Neurocomputing 147 (2015) 160–173

Contents lists available at ScienceDirect

Neurocomputing

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

Intuitive volume exploration through spherical self-organizing map and color harmonization

Naimul Mefraz Khan *, Matthew Kyan, Ling Guan

Ryerson University, 350 Victoria Street, Toronto, ON, Canada M5B 2K3

ARTICLE INFO

Article history:
Received 12 April 2013
Received in revised form
27 August 2013
Accepted 19 September 2013
Available online 19 June 2014

Keywords:
Direct volume rendering
Transfer function
Self-organizing map
Color harmonization

ABSTRACT

Finding an appropriate transfer function (TF) for mapping color and opacity values in direct volume rendering (DVR) can be a daunting task. This paper presents a novel approach towards TF generation for DVR, where the traditional low-level color and opacity parameter manipulations are not necessary. The TF generation process is hidden behind a simple and intuitive spherical self-organizing map (SSOM) visualization. The SSOM represents a visual form of the topological relations among the clusters. The user interacts with the SSOM lattice to find interesting regions in the volume. The color and opacity values are generated automatically from the voxel features based on the user's perception. We also use harmonic colors to present a visually pleasing result. Due to the independence of SSOM from feature type, our proposed method is flexible in nature and can be integrated with any set of features. The GPU implementation provides real-time volume rendering and fast interaction. Experimental results on several benchmark volume datasets show the effectiveness of our proposed method.

1. Introduction

Volume exploration is an important technique to reveal inner structures and interesting regions in a volumetric dataset. However, exploring the volume is a difficult and non-intuitive task since there is no prior information available regarding the data distribution. 3D representation of a volume adds complexity to the whole process. To ease this process, direct volume rendering (DVR) makes use of a transfer function (TF), which maps one or more features extracted from the data (the feature space) to different optical properties such as color and opacity. DVR maps the voxels to color and opacity values directly rather than using some form of surface reconstruction technique. As a result, it can represent the noise or artifacts generated by the typical 3D scanning machines more accurately [1]. Because of this accurate representation, DVR has become a popular technique for volume rendering in magnetic resonance imaging (MRI) and computed tomography (CT).

The TF design is typically a user-controlled process, where the user interacts with different widgets (usually representing feature clusters or 1D/2D histograms) to set color and opacity properties to the feature space. In case of clustering-based TF design techniques, the user can also control some low-level properties like the number of clusters, cluster variance, etc. Most of the recently proposed DVR methodologies [1–4] are based on this philosophy.

However, interacting with the feature space is difficult for the end-user, who may not have any knowledge about feature extraction and clustering. Multi-dimensional feature spaces cannot represent distinguishable properties such as peaks and valleys which are important for proper TF generation from histogram [5]. As a result, the user requires extensive knowledge about the actual volume and the TF widgets themselves. Also, these kinds of widgets try to represent the feature space directly, putting a strict restriction on the type of features used and the dimensionality. Some methods use alternative ways to represent the higher-dimensional feature space through manageable widgets. For instance, in [6], spatial information is encoded in the color values while opacity is derived through intensity and gradient. But these kinds of alternatives are restrictive in the sense that the clustering or histogram generation is not directly derived from the full feature set. Also, only the specific features used in the proposed methods can be used for all datasets. Volume rendering has a wide range of applications in different fields, and one set of features useful for a specific application might be completely irrelevant in another. There is a need to make the method independent so that any feature can be represented to the user in a visual form while maintaining the topological relationship between various data distributions. Hence, a good DVR system should be: (1) intuitive and easy to use without extensive technical experience, (2) time efficient by providing immediate results, (3) have separate classification and feature modules independent of each other so that

* Corresponding author. Tel.: +1 647 9685762; fax: +1 416 979 5280.
E-mail addresses: n77khan@ee.ryerson.ca (N. Mefraz Khan), mkyan@ee.ryerson.ca (M. Kyan), lguan@ee.ryerson.ca (L. Guan).

http://dx.doi.org/10.1016/j.neucom.2013.09.064
0925-2312/

any type of feature can be integrated easily and efficiently and (4) visually pleasing.

A machine learning method that can satisfy all the aforementioned requirements is self-organizing map (SOM) [7]. SOM is an unsupervised clustering algorithm that maps continuous input feature space patterns to a set of discrete output nodes. The nodes are arranged in a pre-defined lattice. Each node has a weight vector associated with it. The input patterns (or feature vectors) are assigned to the closest node in the Euclidean sense. In other words, the nodes "compete" among themselves to be representative of the underlying input patterns [8]. As a result, at the end of the training cycle, the node lattice represents a topology-preserving low-dimensional representation of the original feature space. Because of the unique training procedure of SOM, clusters existing in the original space preserve their relationship in the lower-dimensional space. Because of the regular lattice structure, SOM nodes are easy to visualize. They can also be colored in a very intuitive way by using U-matrices [7], so that the end-user can quickly identify underlying clusters. Since SOM nodes have direct correspondence with the original input data (voxels in this case), the user can be presented with the colored SOM lattice. An end-user can select the clusters he or she deems to be important. In this way, the user will have full control over the regions to be defined as important. Due to the ability to use higher-dimensional features than mere 2D/3D as in the case of widget-based methods, the cluster representation can be sophisticated so that user interest is reflected in the grouping of voxels.

However, mere correspondence between SOM nodes and voxels is not enough for effective volume rendering. We need to generate the color and opacity properties to be assigned to the different clusters selected by the user. The typical trend is to let the user control the color and opacity specifications [6,5]. Especially the color selection is mostly left out as a user task. However, this requires some visualization knowledge which the end user may not possess. In our proposed method, we calculate the opacity and color values automatically from the user selection of voxel groups. The opacity value is generated based on the variance of a group, since variance is related to visibility [9]. For the color values, we take help of *harmonic colors* [10,11]. Harmonic colors are a sets of colors, the internal relationship among which provides a pleasant visual perception [12]. Since our visual experience is largely dependent on the represented colors and their relationships, using color harmonics can result in a much better visual output than using user-defined colors. We also assign the saturation and brightness of each voxel based on its properties. The combination of feature-based parameterizations and color harmonics takes the difficult task of searching through a virtually infinite color space away from the user. In this way, by simply interacting with the SOM lattice, a user can generate the desired rendering quickly and efficiently.

In summary, our proposed system has the following primary contributions:

- Rather than manipulating cluster parameters or optical properties, the user simply interacts with a color-coded SOM lattice representing cluster densities. Due to this visual nature of SOM, there is no need to tweak the cluster parameters and perform operations like split and merge to precisely determine the number of clusters or cluster spread. The user only has to intuitively select or de-select the SOM regions to reveal corresponding structures in a volume.
- The user also does not need to manipulate color and opacity properties. The opacity and color values are automatically generated by combining various voxel group properties with aesthetically pleasing color harmonics. As a result, desired rendering can be achieved quickly and efficiently.
- The proposed model is independent of the type of the features used. As a result, new types of features can be integrated easily and efficiently. Based on the application domain in which the system will be used, the type of feature can be changed. Also, since the feature space is not represented directly like widget-based TF methods, the proposed model is not restricted by 2D/3D features. Higher-dimensional features can also be used.

We use the spherical SOM structure (SSOM) [13–15] because of its restriction-free structure (explained later). The GPU implementation of our method provides fast interaction with the SSOM and real-time volume rendering.[1]

The rest of the paper is organized as follows: Section 2 discusses the related works in TF and SOM. Section 3 details our proposed method. Section 4 provides results on some well-known datasets. Finally, Section 5 provides the conclusion.

2. Related work

The relevant works fall into two categories: (1) transfer function specification and (2) spherical self-organizing maps.

2.1. Transfer function specification

Traditionally, TFs were one-dimensional, where the color and opacity are derived from intensity value only. The user assigns different color and opacity values to the 1D histogram derived from intensity values [17]. However, volume data is typically complex in nature and naturally, the intensity value alone is not a good feature to separate voxels into different groups. Multi-dimensional TFs make use of other features along with intensity values.

The work of Kindlmann and Durkin first popularized multi-dimensional TF [1]. Their work used gradient magnitude alongside intensity value to build a 2D histogram, where material boundaries in a volume can be identified as arches. The arches can be assigned color and opacity values manually with the help of the widgets. The limitation of this method was the overlapping of arches for different boundaries, making them indistinguishable. Their later work [18] has included the second derivative to build the histograms, however, the problem was not solved properly. In [19] the *LH-Histogram* was proposed. The LH-histogram is calculated by tracing the voxels along the boundary. The low and high intensity values along the gradient were recorded and used to build a histogram. In this approach, boundaries are represented as blobs rather than arches, which are relatively easier to distinguish [19]. The main drawback with this approach is that the histogram takes a long time to calculate [19].

Curvature-based approaches are also popular for volume rendering. In [20], a contour spectrum of isosurfaces is built using scalar and contour attributes. The user can select different isosurfaces and assign visual properties to them. Another method using histograms built from curvature measurements was proposed in [21].

Recently, *feature size* has also been used to design multi-dimensional TFs. In [22], the voxels were assigned a scale field, which is calculated from the relative size of a particular feature for each voxel. A size-based distribution is then presented to the user for TF generation. A region-growing technique was used to find the feature size in [23]. In [24], structure sizes were estimated based on a user-defined intensity range. This structure size was

[1] The proposed method is an extension of our work presented in the 9*th Workshop on Self-Organizing Maps* [16].

then used to generate a structure size enhanced (SSE) histogram, which in turns was used to specify the TF.

Users generally may not be able to focus on voxels too far apart due to the complexity of volume datasets. As a result, spatial information has been used as features in some recently proposed methods. Local histograms combining intensity and spatial information in the local neighborhood were proposed in [25]. In [26], the α-histogram was proposed, which incorporates the property of spatial coherence to produce a global histogram. Spatial information was combined with the 2D histogram in [6]. A distance-based TF was introduced in [27] which assigns optical properties to structures based on the distance to a selected reference structure in order to hide or emphasize structures. A volume histogram stack (VHS) is introduced in [4], where the histogram is created by aligning the histograms of the different slices and incorporating spatial information with them. Then a self-generating hierarchical radial basis function network is used to generate the TF. Self-organizing maps were used in [28] to perform dimensionality reduction on the feature space. Different features of the voxels and the maps were then linked to the color and opacity values to generate the final rendering based on user action.

Another alternative for TF specification is the *image-centric* approach, where the users are given the option to interact with the data in the image domain i.e. directly with the output [29]. The changes to the TF are done in the background. Image-centric approaches also include methods where the TF is picked by the user from a set of TFs based on the output rendering, not based on volume properties. The design gallery method [30] is a popular image-centric approach. In this method, several different TFs are generated by varying the input parameters. The user is then presented with the output generated from all the TFs, and the task of final TF selection is left to the user. The method in [31] uses Gaussian mixture models (GMM), where the user can manipulate different parameters of the models instead of directly manipulating TF parameters. An intelligent user interface has been introduced in [32], where the user can paint on different slices to mark areas of interests. Neural network based techniques are then used to generate the TF. A spreadsheet-like interface based on Douglas–Peucker algorithm is presented in [33].

Finally, some recent methods have used different clustering algorithms on the histogram data to divide the voxels into different groups and assign TF properties accordingly. Voxels were grouped based on their LH-histogram and spatial information in [34]. A non-parametric TF automation method based on the kernel density estimation was proposed in [5]. The estimated density distribution is divided into different ranges, and the user specifies the color and opacity values for each range to generate the final output. A parallel technique based on mean-shift clustering of voxel intensities and spatial values was proposed in [35]. A semi-automatic method based on a combination of mean-shift clustering and hierarchical clustering was proposed in [2], where the user has control over the hierarchical clustering results.

Zhou and Takatsuka were the first to use the concept of color harmonization in TF generation [11]. Their method adopted a residue flow model based on Darcy's law for TF generation. Color harmonization was also used in [9], where the K-means++ algorithm was used to divide the voxels into groups based on their intensity values and spatial information. This method also uses automatic assignment of opacity values based on the user's perception.

Despite all these efforts, TF specification is still a difficult task. Most of these methods still rely on extensive user interaction for histogram manipulation or some form of cluster control (e.g. number of clusters, variance of clusters, merging and splitting of clusters) in the feature space. An expert from medical or architectural background might not be familiar with these specifications. A completely automated rendering is also not very desirable, as only the user has the necessary domain knowledge to decide which regions in the volume he or she needs to focus on. Hence, the user control over the rendering is necessary, but it also needs to be as simple as possible. Moreover, most of these methods present visualization of the feature space itself. Hence, it is not possible to incorporate new features. As volume data can be complex, noisy and highly domain dependent, a straight-forward way to incorporate new features into the system is necessary. Our proposed model eliminates these limitations by using self-organizing maps (SOMs).

Our method draws some inspiration from the method proposed in [28]. However, the SOM visualization and volume exploration in our proposed method are completely different. Our method presents the visual representation of the cluster densities and allows the user to find the correspondence between SOM nodes and voxels interactively. On the other hand, the SOM is used in [28] mainly for dimensionality reduction. The Gaussian TF generation in [28] can also be difficult to control, as the direct correspondence between voxels and SOM nodes are not fully exploited. We also retain spatial information in our feature set (Section 2.1), which can produce better separation of voxel clusters. Moreover, our method uses color harmonics to generate visually pleasant renderings rather than relying on ad-hoc color assignment. Lastly, in [28], a two-pass SOM training is used to better represent the boundary voxels, which can slow down the training process. Instead, we use the count-dependent parameter from [13] (detailed in Section 3), which can result in faster training times. We also use the spherical self-organizing map for its advantages as described in the next section.

2.2. Spherical self-organizing maps

The self-organizing map (SOM) is an unsupervised clustering method proposed by Kohonen [7] that clusters the data and projects data points onto a lower-dimensional space while preserving the input topology. The data points are projected onto a regular lattice during training, when the weights of the nodes in the lattice are updated. In this way, after training the initial lattice "self-organizes" itself to represent the data distribution. Different coloring methods are then applied on the lattice to color code it so that the cluster regions can be visualized. As stated before, the topology-preserving clustering and easy visualization properties of the SOM make it an attractive choice for TF generation.

The underlying lattice structure is primarily responsible for the visual representation of the SOM. The SOM learning occurs due to lateral interactions among nodes in the underlying lattice. A discontinuity in the lattice will result in a restricted neighborhood and can lead to inefficient learning [14].

The most popular lattice structure is the 2D SOM, where the nodes are represented by a sheet of connected nodes (Fig. 1(a)). Since the SOM training is based on the lateral connections among the nodes, a 3D structure provides an additional dimension of connections and therefore offers increased learning as compared to a 2D lattice [13]. Commonly used 3D SOM consists of the cube lattice structure (Fig. 1(b)).

However, all these structures have a restricted neighborhood problem. In [13] it has been shown that the 2D SOM fails to accurately approximate a point cloud due to its open boundaries. It has also been shown that the 3D cubic structure has a rather "forced" continuity, which results in an unwanted folding effect. These problems were perceivable due to the use of 3D point clouds in [13]. But for higher-dimensional data these kinds of training errors will be difficult to detect.

An optimum lattice structure should offer minimum topological discontinuity, as is the case of a spherical structure. The spherical lattice is created by subdividing an icosahedron to obtain

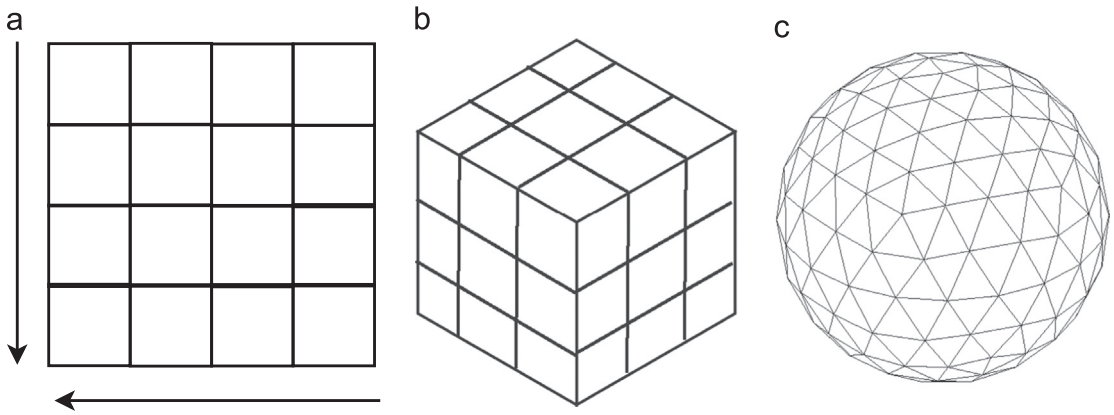

Fig. 1. Depiction of different SOM lattices. (a) 2D (open boundaries), (b) cube ("forced" continuity), and (c) spherical (continuous).

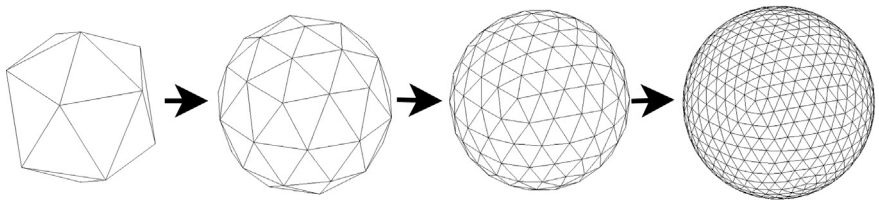

Fig. 2. The first three levels of subdivision of the basic icosahedron.

a uniform distribution of nodes resulting in a structure with an overall symmetry and continuity. From the basic icosahedron, higher resolution lattices can be created by bisecting every edge of the icosahedron and radially projecting the new vertex to lie on a sphere of unit radius [14]. All the new vertices will still maintain the equilateral triangular structure of the basic icosahedron. The first three levels of subdivision of the basic icosahedron are depicted in Fig. 2. In this way, the lattice will have a continuous structure with minimum discontinuity among nodes, which can result in better separation among the clusters.

There are several variations of the spherical lattice depending on the underlying data structure used to represent the sphere and the visualization technique used, such as the hyperbolic SOM [36] or the Geodesic SOM [37]. We have used the version introduced in [14] mainly because of it being visually more intuitive and easier to perceive [13].

3. Proposed method

In this section, we describe the details of our proposed system based on the spherical SOM (SSOM) structure. This section is divided into three subsections where we discuss different aspects of the proposed system: (1) feature extraction, where we discuss about the features used in this paper to model a volume for TF generation; (2) SSOM training, where the detailed steps of the spherical SOM training are described; and (3) TF representation and exploration, where we discuss our opacity and color assignment for voxel groups based on color harmonization and the user's perception. We also discuss in detail how simply and efficiently the user can explore our color-coded SSOM lattice to find interesting regions and group voxels together. The color and opacity values are generated accordingly with immediate rendering results through GPU implementation.

3.1. Feature extraction

As described in Section 2.1, multi-dimensional TFs mostly use some form of histogram based on intensity and gradient magnitude. However, one problem with histogram is that it cannot retain the spatial information present in a volume [6]. As present day scanning systems for volume data (CT, MRI, etc.) have some inherent noise associated with them, losing spatial information might prove to be costly and may not cluster the volume data in a suitable way for volume rendering. On the other hand, incorporating the spatial information directly is difficult for any histogram-based approach, as the histogram will be even higher-dimensional and there is no easy way to represent it visually. As a result, in most histogram-based approaches, the spatial information is used for color generation only [6]. However, as described before, our proposed model can incorporate higher-dimensional features due to the use of SSOM lattice. As a result, in our proposed model, the spatial information is directly embedded into the feature definition. To emphasize the boundaries between materials [1], we also use the intensity value of each voxel, its 3D gradient magnitude and the second derivative (obtained through eigenvalues of the Hessian matrix), all of which are popular features for traditional histogram-based TF design [1]. Our 6D feature set for each voxel consists of the following features:

- The X, Y and Z coordinates.
- The intensity value.
- The 3D gradient magnitude. The 3D gradient magnitude for each voxel is defined by

$$G = \sqrt{G_x^2 + G_y^2 + G_z^2}, \tag{1}$$

  where G_x, G_y and G_z are the gradient values along X, Y and Z directions, respectively.
- The second derivative. This feature can be calculated from the eigen values of the Hessian matrix [38]. The Hessian matrix can be defined as

$$H = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}, \tag{2}$$

  Where I_{xx}, I_{xy}, …, I_{zz} are the second partial derivatives. The sum of the three eigenvalues of this matrix $\lambda_1+\lambda_2+\lambda_3$ (where λ_i

represents one eigenvalue) roughly approximates the information obtained from the second derivative of the intensity values [38]. This sum value is used as a feature in our experiments.

All the features are scaled to fall in the interval of {0,1}. Ideally, we would still like to emphasize the boundaries of materials over spatial similarity. Hence, the used features are weighted. For our experiments, we use a weight set of {0.5,0.5,0.5,1.5,3,2} for the aforementioned features. These values were picked by a trial and error method. However, minor changes to these values do not have any adverse effect on the results.

3.2. SSOM training

The training phase of the SSOM is similar to the classical SOM [7]. Let the input space of N voxels be represented by $\mathcal{X}=\{\mathbf{x}_i\}_{i=1}^{N}$. Let the SSOM be represented by M nodes ($M \ll N$). Each node in the SSOM lattice has a corresponding weight vector $\mathbf{w}$. All these weight vectors together represent our SSOM space $\mathcal{W}=\{\mathbf{w}_i\}_{i=1}^{M}$. Each node also has a *neighborhood* associated with it. A neighborhood is a set of nodes consisted of the node itself and its neighbors. Let the neighborhood set for node i be represented by Θ_i^r. Here, r represents the neighborhood spread, $r=1,\ldots,R$. R is the maximum neighborhood radius, which is set to a value such that it covers half of the spherical space [13].

The input voxels are randomly introduced to the SSOM during training. For each voxel, a *Best Matching Unit* (*BMU*) among all the nodes is selected. BMU is the node which is closest to the input voxel according to some similarity measure. Euclidean distance is used as a distance measure. The update step then takes place, where the weight vector of the BMU and its neighboring nodes (Θ_{BMU}^r) are updated in such a way that they are pulled closer to the weight of the input voxel. After training, the SSOM weight vectors are arranged in such a way that they represent the underlying distribution of the input data (the voxel features in this case).

Besides using the traditional SOM parameters for training, a count-dependent control over how the winning node is being chosen is also used in our system [13]. This parameter is inspired from *frequency-sensitive competitive learning* (FSCL) [39]. FSCL addresses the problem of nodes of the lattice being under-utilized by introducing a measure of how frequently a node is being selected as the BMU. Each node has a count c_i associated with it (see the *weight update* step below), which increases when the node wins. This count in turn is used with the distance measure to "penalize" the winning node i.e. increase its distance measure for the next input vectors [39]. This is especially necessary in our case because in a volume, typically there is almost 80–90% homogeneous regions. But the other 10–20% is usually more important, since they are the boundary voxels. Without a count-dependent control, the boundary voxels will be assigned to only a few winning nodes frequently. As a result, eventually homogenous regions will take over the whole map and important regions (boundaries) will not get enough map space to be noticed. In the Experimental results section, we provide an analysis of the effect of count-dependent parameter on the clustering and the eventual rendering of a dataset.

The step-by-step training algorithm is described below:

- *Initialization:* The weight vectors of the SSOM nodes are initialized first. Random values can be used for initialization, but as pointed out by Kohonen et al. [7], random initialization will take more time to converge. We have followed the initialization method stated in [7] i.e. initialize the weight vectors with values that lie on the subspace spanned by the eigenvectors corresponding to the two largest principal components of the input data distribution. As explained above, a count vector $\mathcal{C}=\{c_i\}_{i=1}^{M}$ [13] is used to keep track of the hits to each node. This vector is initialized to zero. This is used in the BMU selection step to prevent cluster under-utilization.
- *Training:* For each input voxel $\mathbf{x}$, do the following:
  - *BMU selection:* Calculate the Euclidean distance of the voxel feature vector $\mathbf{x}$ with all the nodes as follows:
    $$e_i=(c_i+1)\|\mathbf{x}-\mathbf{w}_i\|, \quad i=1,\ldots,M. \tag{3}$$
    BMU is the node for which this distance is the smallest.
  - *Weight update:* Update the weights for the BMU and its neighboring nodes (defined by Θ_{BMU}^r) as follows:
    $$\mathbf{w}=\mathbf{w}+b(t)*h(s,r)*\|\mathbf{x}-\mathbf{w}\|, \tag{4}$$
    $$c_{\mathbf{w}}=c_{\mathbf{w}}+h(s,r), \tag{5}$$
    where $\mathbf{w}\in\Theta_{BMU}^r$, $b(t)=\alpha e^{-t/T}$ and $h(s,r)=e^{-r^2/s*R}$.
    The functions $b(t)$ and $h(s,r)$ control the rate of learning and the neighborhood effect, respectively. $b(t)$ decreases in value as the iteration number $t=1,2,\ldots,T$ increases. It also depends on the learning rate α. $h(s,r)$ depends on the neighborhood size parameter s, which is user controlled. $h(s,r)$ is a Gaussian function. The further a neighboring node is from a BMU, the less its weight will be affected.
    As discussed before, the count-dependent parameter $c_{\mathbf{w}}$ is increased here to prevent cluster under-utilization. Observing Eqs. (5) and (3), we can see that the BMU and its neighbors are penalized for winning by increasing the count. In the next update step, the distance measure with these nodes will be higher due to the increase of count. This ensures that one node (or its neighbors) does not win too many times, and the entire map is utilized [39].
- Repeat the training steps for a pre-defined number of iterations (T).

The main control parameters in SSOM training are the learning rate α, the number of iterations T and the neighborhood size parameter s. In case of volume rendering, we are dealing with a huge number of voxels (e.g. for a volume of dimension 256 × 256 × 256, there are over 16 million voxels). Among these voxels, only around 10–20% are boundary voxels, the others represent mostly homogeneous regions. For the voxels belonging to homogeneous regions, the features are very similar. As we will show in the Experimental results section, for such high number of voxels with similar features, the SOM typically stabilizes reasonably within only 1–2 iterations. The α is set to 0.1 in our experiments. The neighborhood size parameter is set to $s=2$, which is determined through trial and error.

3.3. TF representation and exploration

After training of the SSOM is completed, the weight vectors associated with the nodes of the SSOM represent the underlying clustering of the voxels. We map this SSOM to a suitable TF in three steps: (1) present a color coded graphical representation of the SSOM, (2) provide the user with interaction options to select group interesting regions in the SSOM together, and (3) map selected regions of the SSOM to a suitable TF and show the rendering of the volume with the generated TF. Due to our GPU implementation, the user can see the rendering of the volume in real-time while interacting with the SSOM.

To color code the spherical lattice, the U-matrix approach is used [13], which provides a visual form of the distance between the weight vectors of the nodes. For each node, the average Euclidean distance of its weight vector with the weight vectors

of all the immediate neighboring nodes is calculated. These distance measures are then mapped to a color-map for visualization purposes. In this way, a homogeneously colored region will represent a cluster, while the cluster boundaries will incur a change in coloring. An important point to note here is that since volume rendering is an entirely perceptual process, it is not important to strictly define how many clusters we have or whether the cluster boundaries are very well defined or not. The important point is to color the SSOM lattice in such a way that intuitively interacting (explained below) with it will directly result in meaningful rendering. Representation of cluster densities in the form of color-map through U-matrix serves this purpose.

Our target is to keep the user interaction as minimum and efficient as possible. However, only the user has the domain knowledge to know what is important in the volume dataset. Therefore, we only leave the selection of voxels to the user. In our experiments, we use the third-level subdivision of the basic icosahedron (Fig. 2) which has 642 nodes. For a typical volume, usually the user is interested in only a few groups (2–3) of voxels. For example, a CT scan primarily retains the bone structures. The user will ideally be interested in grouping the voxels based on different bone densities or regions. All our SSOM nodes correspond to some number of voxels (the voxels for which the node was selected as BMU). The U-matrix based coloring represents the existence of potential clusters, but we leave the final decision of grouping the nodes and defining them as a single group to the user. We provide the user with the control to group the SSOM nodes together to define different *group of voxels*. The user performs this by dragging a rubber-band control on top of the visualized SSOM lattice. The user is immediately presented with an intermediate rendering based on basic pre-defined colors to provide a rough idea of which voxels were selected as a result of his or her action. When the user is satisfied with the selection, he or she can define the current selection as one group, and do further selections for the next group. As soon as a group is defined, the rendering is updated to show the final result based on the properties of the voxels in the group and color harmonization. After a group is finalized, the visualization for it can be turned on/off temporarily through the user interface, so that the user can do further node selections without any hindrance.[2]

For the intermediate rendering, since we do not have a definition of a group yet, we simply map the color channels to different features. One channel is mapped to intensity, while the other two channels are mapped to the 3D gradient magnitude. Note that for our original method proposed in [16], this intermediate rendering was actually the final result, as the user did not have an option to define different groups. This put a limitation on the types of rendering that can be achieved. With several user-defined groups, we can assign visually more pleasing color and opacity values based on the user's perception and color harmonization as described below.

3.3.1. Color and opacity assignment

After the user defines the separate voxel groupings on the SSOM lattice, we generate the color and opacity values to create the final rendering. We have put special attention to the color selection of different groups. In most of the existing methods, the colors are often determined by ad-hoc assignment or personal preference [9]. Since volume exploration is a lengthy and complex process, visual aesthetics is very important. The contrast among the colors is also important to differ among different types of materials (voxel groups in our case) efficiently. Hence, we apply the theory of color harmonization [12] for our color assignment. Harmonic colors define sets of colors that are aesthetically pleasing to the user [12]. In [40], a color wheel was introduced in which color harmony is described across color hue values. A harmonic set is described by specifying the relative position of the colors on the wheel rather than specific color themselves. The harmonic colors are defined based on user evaluations rather than strict mathematical formulations. Based on this color wheel, several templates can be defined. Fig. 3 shows the four templates that we have used in our method, namely type V, type L, type T and type X. All these templates can be rotated at arbitrary degrees to generate a new set of colors (refer to [10] for the detailed specifications on these templates).

We use these templates to generate the hue values for *HSV* color space. We need separate hue value for each group of voxels selected by the user. We equally space the required hue values along the available hue range so that we have a visually pleasing result as a combination of all of them.

However, if we simply use one hue value for each user selected group, the details will not be revealed. For volume rendering, boundary enhancement is important to reveal the inner structures [1]. As a result, instead of using one hue value, we use a small range with one lower and one higher limit, ξ_l^i and ξ_h^i for each group. For our experiments, the difference between ξ_l^i and ξ_h^i is 5% of the whole hue wheel (or 18°). This still gives us enough hue range to space out all the groups (typically 2–3 as described before) reasonably around the hue templates.

The hue value for each voxel of each group is calculated by the following equation:

$$H_i^v = \xi_l^i + (\xi_h^i - \xi_l^i) * F_v, \quad i = 1, \ldots, Z, \tag{6}$$

where Z denotes the total number of groups selected by the user. F_v is the *gradient factor* and is defined as

$$F_v = \frac{G_v}{\max_{v \in \mathcal{Z}} G_v}. \tag{7}$$

Here, $\mathcal{Z}$ denotes the group that the voxel v belongs to. G_v is the 3D gradient magnitude for each voxel (defined in Eq. (1)). In this way, even for a single group, the hue values will have some variation based on the gradient value instead of having a single hue value assigned to all of them.

In the *HSV* color space, the S and V values are also very important to define a color. The S and V values respectively determine the saturation and brightness of the color. We generate the S and V values based on the understanding of the user's perception [9]. Voxel groups that have a small spatial variance occupy a smaller viewing area compared to groups that have relatively larger spatial variance. We can intuitively assume that the groups with smaller spatial variance needs to have a more saturated color to highlight them properly [9]. Hence, we calculate the S value for each voxel group according to the following equation:

$$S_i = \frac{1}{(1+\sigma_i)}, \quad i = 1, \ldots, Z. \tag{8}$$

Here, σ_i represents the spatial variance of each voxel group.

Similarly, since the rendering results are usually viewed at a distance from the whole volume, we can assume that voxel groups closer to the center of the volume need to be brighter so that they are not overshadowed by groups that are further from the center [9]. Hence, the V value for each voxel group is calculated as follows:

$$V_i = \frac{1}{(1+\mathbf{D}_i)}, \quad i = 1, \ldots, Z. \tag{9}$$

[2] The reader is encouraged to see the video demo at http://goo.gl/Z4NJD for a better understanding of the whole process.

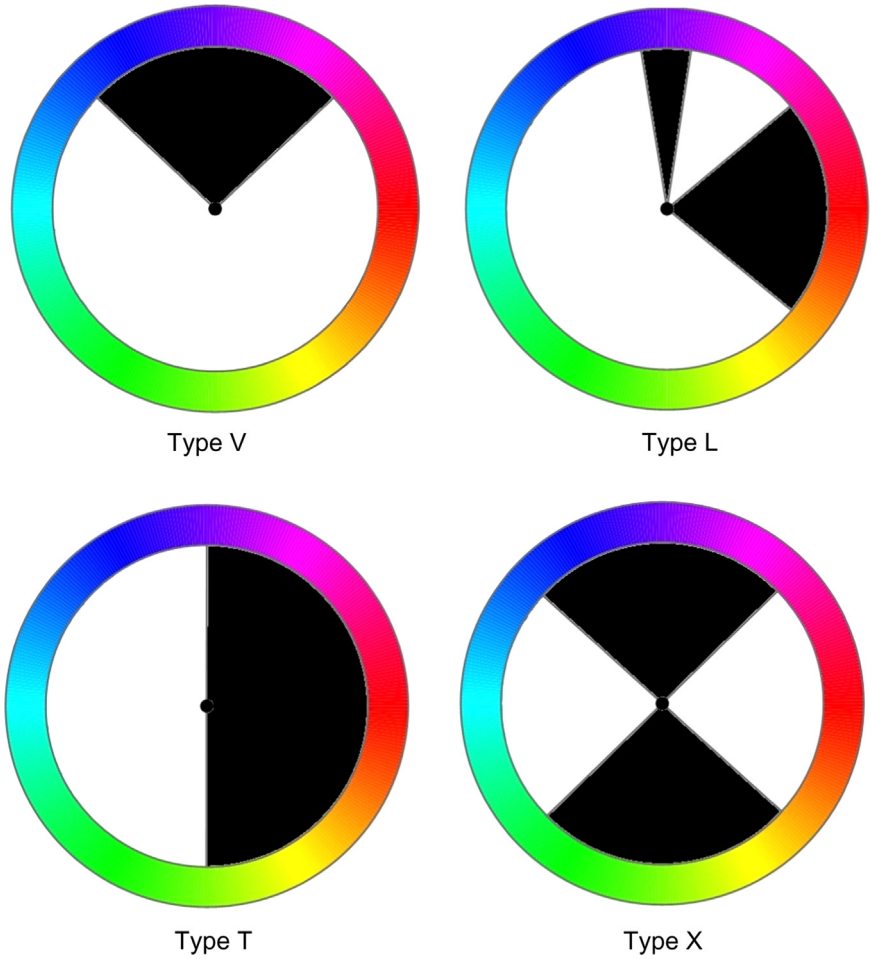

Fig. 3. Hue templates.

Here, $\mathbf{D}_i$ denotes the distance of the centroid of the ith group to the center of the volume [9]. By assigning the saturation and brightness values this way, we can assign more vivid colors to the voxel groups that are relatively more difficult to highlight.

The last parameter to define the full TF is opacity. Since voxel groups with smaller spatial variances are likely to be obstructed for proper viewing by groups with larger spatial variances, we calculate the opacity values based on spatial variances. We also want to emphasize the boundaries to reveal the detailed structures. Hence, our opacity values for each voxel are calculated using the following equation:

$$O_i^v = \left(1 - \frac{1}{\sigma_i}\right) * (1 + F_v), \quad i = 1, \ldots, Z. \tag{10}$$

Here, σ_i is the spatial variance of the i-th voxel group. F_v is the gradient factor as defined in Eq. (7). Inspecting Eq. (10), we can see that voxel groups with smaller spatial variances will be assigned higher opacity values. Moreover, all the opacity values will go through boundary enhancement by being multiplied by $(1+F_v)$.

After calculating the *HSV* and opacity values, we convert them into the final *RGBA* texture. The *HSV* triplets are converted into *RGB*, and the opacity value is appended to generate the *RGBA* quadruples. This is passed to the rendering stage to generate the final rendering.

For the rendering stage, we use the Visualization Toolkit (VTK) [41]. The *RGBA* texture passed to VTK is rendered in GPU. As a result, the user can see the rendering changes in real time while selecting and assigning voxels into different groups.

4. Experimental results

In this section, we provide detailed analysis and results of our proposed system on several volume datasets. We used five volume datasets [42], CT scans of a Foot, the Visual Male Head dataset, the Carp dataset, the Engine dataset and the Lobster dataset. Generally, volume datasets have a lot of vacant regions (air around the object). If we build the SSOM directly on the dataset, these regions will generate a misleading clustering. A simple region growing algorithm [43] can be used to separate the object from these regions first, where a voxel in the vacant region can be used as seed to separate the object from the background. However, the voxels in the vacant region typically have zero (or very close to zero) intensity values. Hence, a simple thresholding with a low intensity value can separate the background. In practice, the region growing option provides marginally better separation. But the thresholding method is several times faster than the region growing approach. Hence, in our experiments we opted for the thresholding method to separate the object from the vacant regions. These separated voxels were then fed to the SSOM training algorithm.

After the SSOM training, the user is presented with a visual form of the spherical lattice color coded with the U-matrix approach as described before. The user can then perform selection/de-selection on the lattice, and the corresponding voxels are shown with an intermediate rendering. When the user is satisfied with the selection, he or she can assign the current selection as a group of voxels. This group is then assigned the color and opacity values according to the equations described in Section 3.3.1 for

final rendering. The user can then move on and select new group of voxels. At each step of group definition, the rendering is updated to reflect the new color and opacity values. To see how simple and intuitive the whole process is, we highly recommend the reader to see the video demo (located at http://goo.gl/Z4NJD). The video demo shows a brief walk-through of the whole system, especially emphasizing on the effectiveness of our color and opacity assignment techniques.

Fig. 4 shows intermediate renderings of all the datasets when no grouping is performed (i.e. all the nodes are treated as a single group). Note that the color assignments for these intermediate renderings are arbitrary. The color channels are mapped to different features and scaled to fall at random color regions to generate these renderings. These renderings still use the coloring concept we have used in [16], which is not very helpful when complex features are being sought after. These renderings are presented here to provide the reader with an idea on how these datasets might look without any voxel grouping or color processing. As we will show subsequently, our new color and opacity assignment based on color harmonization and the user's perception can reveal the details easily and efficiently.

First, we demonstrate the effect of the count-dependent parameter defined in Section 3.2. As described before, due to the dominance of homogeneous regions in typical volume datasets, we need to make sure that the boundary voxels are getting enough map space. Hence, the count-dependent parameter is used so that cluster under-utilization is eliminated and boundary voxels are not overshadowed by other voxels [39].

Fig. 5 shows the intermediate renderings of the voxels corresponding to the user selection, both with the count-dependent parameter and without it. As we can clearly see, without the parameter, the voxels representing the inner structure of the bones are not converged into a cluster properly. Other homogenous voxels also become part of the selection and the rendering is unclear. By using the count-dependent parameter, the voxels can be grouped in a much easier way, and the user selection clearly corresponds to the inner structures. For this reason, we used the count-dependent parameter in our training algorithm.

Next, we show the empirical convergence behavior of the SSOM with the Foot dataset. As stated before, typical volume datasets have a small number of (around 10–20%) boundary voxels. The other voxels belong to homogenous regions and have similar feature characteristics. As a result, the SSOM stabilizes very quickly, within 1–2 epochs. Perfect convergence is also not necessary for our system, as the final result is visual rather than quantitative.

Fig. 6 shows a plot of the change in the weight vector as the number of epochs increases [13]. This experiment was performed on the Foot dataset. The change in the weight vectors is obtained as a scaled distance between the current weight vectors and the previous one. As we can see, after Epoch 1, the change in weight vectors is not significant. The map does not necessarily converge,

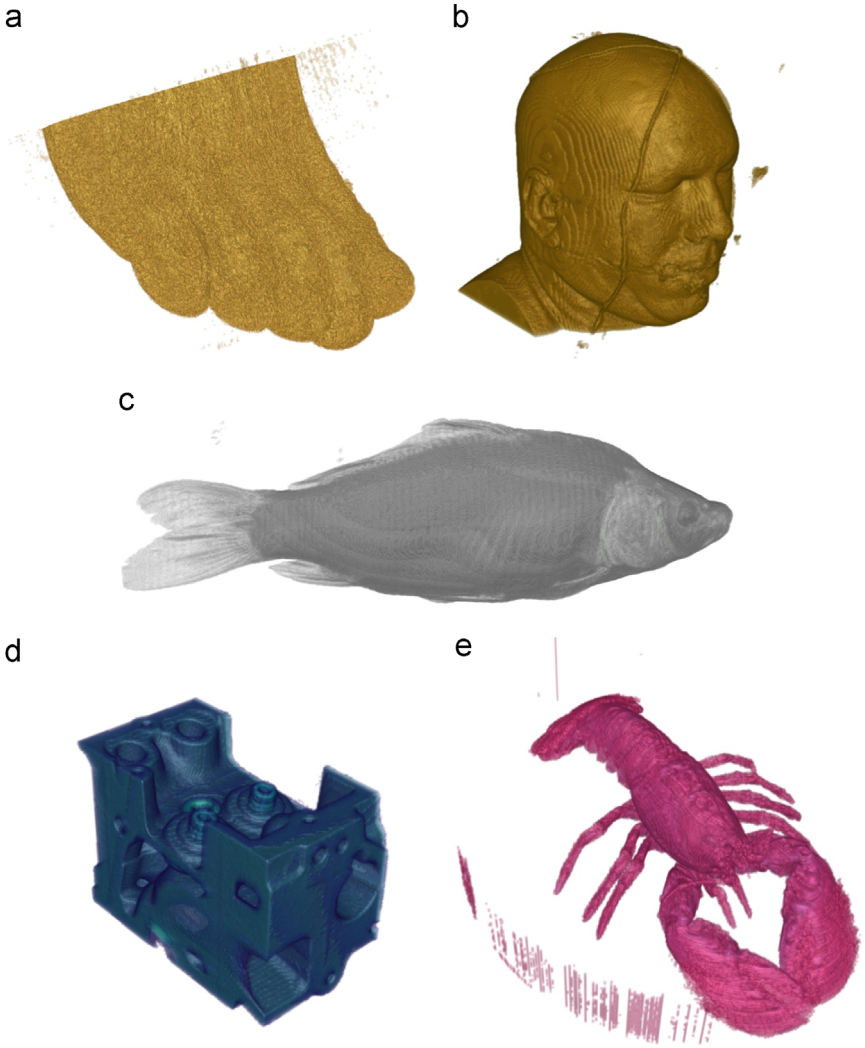

Fig. 4. Intermediate rendering of all the datasets without any voxel grouping or color harmonization. (a) Foot. (b) Visual Male Head. (c) Carp. (d) Engine. (e) Lobster.

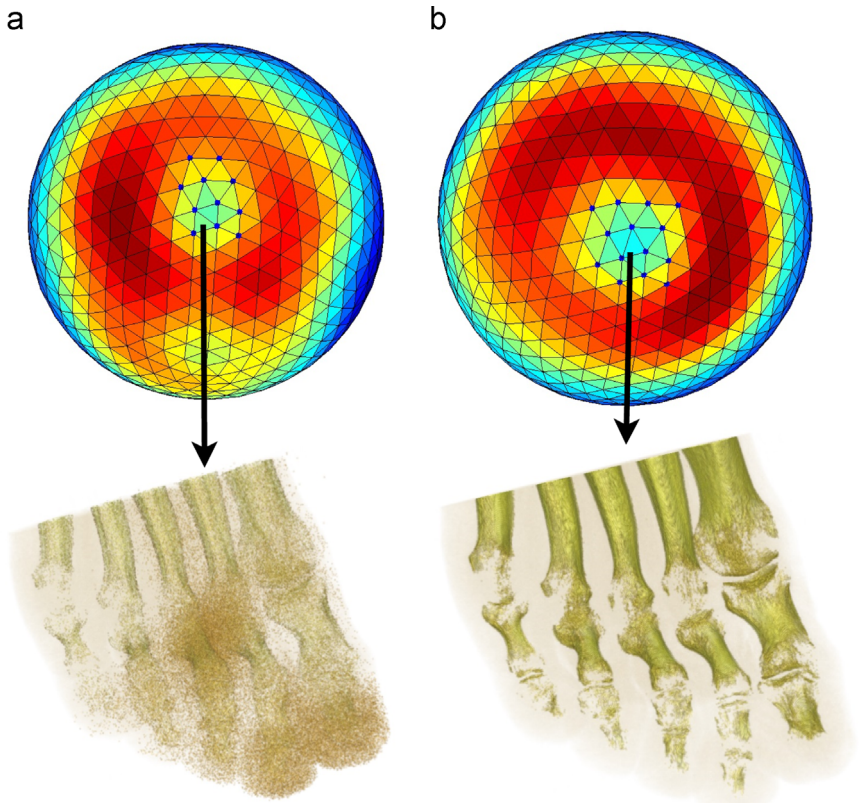

Fig. 5. Intermediate rendering of the Foot dataset corresponding to user selection when the training is performed (a) without the count-dependent parameter and (b) with the count-dependent parameter.

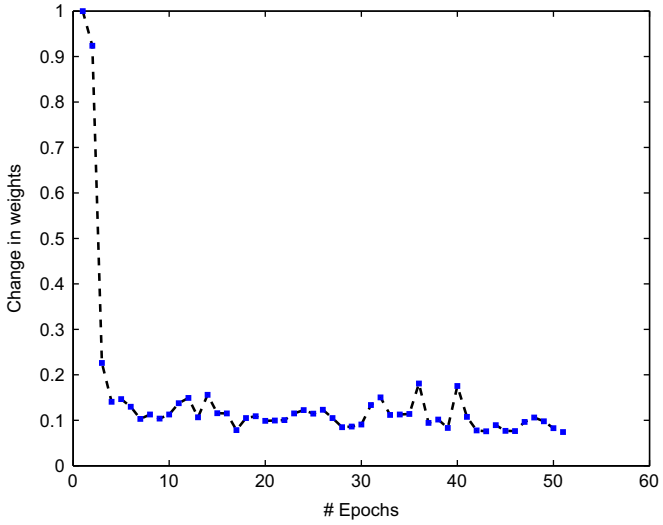

Fig. 6. Convergence characteristic of the SSOM for the Foot dataset.

but we can say that the map is "stable" enough for our system, as the changes are only minor after the first iteration.

Fig. 7 show the visualization of the SSOM lattice for the Foot dataset after one and five epochs. As we can see, even visually the maps are almost similar with minor differences. This again justifies our claim that 1–2 iterations of the training algorithm is good enough for the stabilization of the map.

Finally, we show the usefulness of our system in action. Fig. 8 shows the final render results of different voxel groups defined by the user. The voxel groups can be seen on the spherical lattice, with different groups of nodes separately colored (white, blue and black dots). As can be seen, due to high-dimensional feature clustering, the structures in the dataset can be revealed quickly and efficiently. Group 1 (corresponding to the nodes colored in white) reveals the inner structures of the bones, while Group 2 (corresponding to the nodes colored in blue) emphasizes the joints among the bones. All the other voxels are assigned to Group 3 (corresponding to the nodes colored in black), which roughly represents the outer layer of the foot.

The effectiveness of our color and opacity assignment techniques can be seen in Fig. 9. Fig. 9(a) shows Group 1 and Group 2 together, while Fig. 9(b) shows all three groups together. As can be seen, due to using color harmonization, all these groups are distinguishable even when they are rendered on top of each other. For this dataset, we used the T-type template (Fig. 3) with default setting i.e. starting at 0°. Since our opacity assignment depends on the spatial variance (Eq. (10)), the inner groups of voxels are assigned higher opacity values (less transparent). Moreover, due to the use of boundary enhancement with gradient factor, the resulting renders are revealing the structures clearly.

Another interesting observation is the spread of the voxel groups selected by the user. In the map, the blue colors suggest the existence of a cluster, while a shift towards red denotes change of cluster. However, as we can see from Fig. 8, Group 2 is defined around group 1 and on a red patch in general. Ideally, a perfect combination of SOM algorithm and feature set should separate the clusters very clearly. But as we stated before, volume rendering is a complex process and the principal quality assessment is how the user perceives the output. If we picked the voxel groups automatically based on the cluster density, a desirable output will be difficult to achieve. But by leaving the final group definition as a user task, we provide the user with enough flexibility, but hide the complexity of feature extraction, voxel correspondence, color and opacity assignments by making them automated. The only knowledge the user needs is the detection of a small cluster.

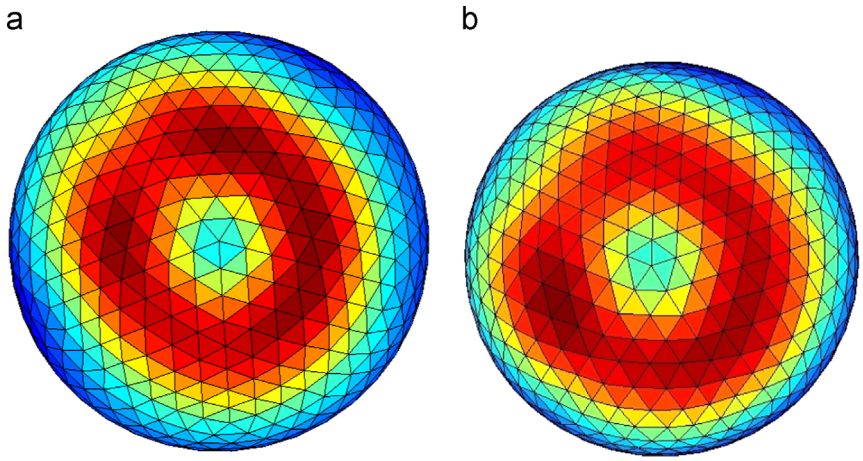

Fig. 7. SSOM visualizations for the Foot dataset (a) after one epoch and (b) after five epochs.

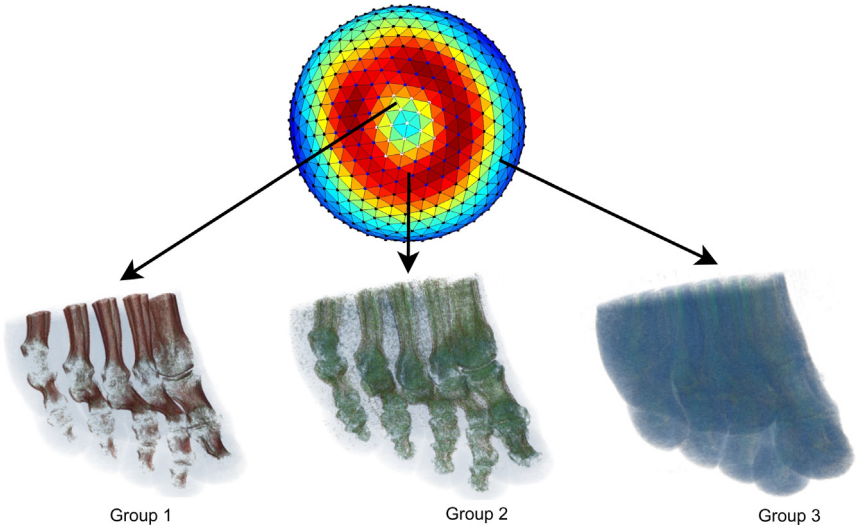

Fig. 8. Different user-selected voxel groups and corresponding renderings of the Foot dataset. White, blue and black dots refer to Group 1, Group 2 and Group 3 (pointed out by the arrows), respectively. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

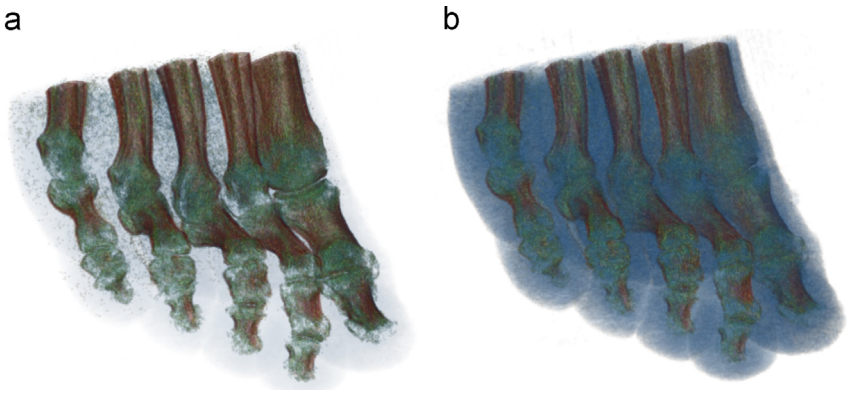

Fig. 9. Rendering of the Foot dataset (a) Group 1 and Group 2 together. (b) All groups together.

Typically, a small cluster (blue patch) surrounded by a shift towards other clusters (red patches) suggests that a small group of voxels reside in this cluster which are different from other regions of the volume. So the user can work his/her way out from there and define the voxel groups as he or she wants to see it. With this flexibility, our system provides a robust platform to generate different kinds of renderings from a single dataset.

Fig. 10 shows the results from out next dataset, the Visual Male head. X-type template with default setting (i.e. starting at 313.2°) was used for this dataset. Here, we see that there are two user-defined voxel groups. Group 1 (corresponding to the nodes colored in white) reveals the detailed skull structure. Group 2 consists of the other voxels to represent the outer layer (corresponding to the nodes colored in black). Here, the skull structure is of interest, which can be separated very easily through our system. Fig. 11 shows the two groups on top of each other. Again, as we can see, due to our intelligent color and opacity assignments, the outer layer is more transparent and the nested features are still visible.

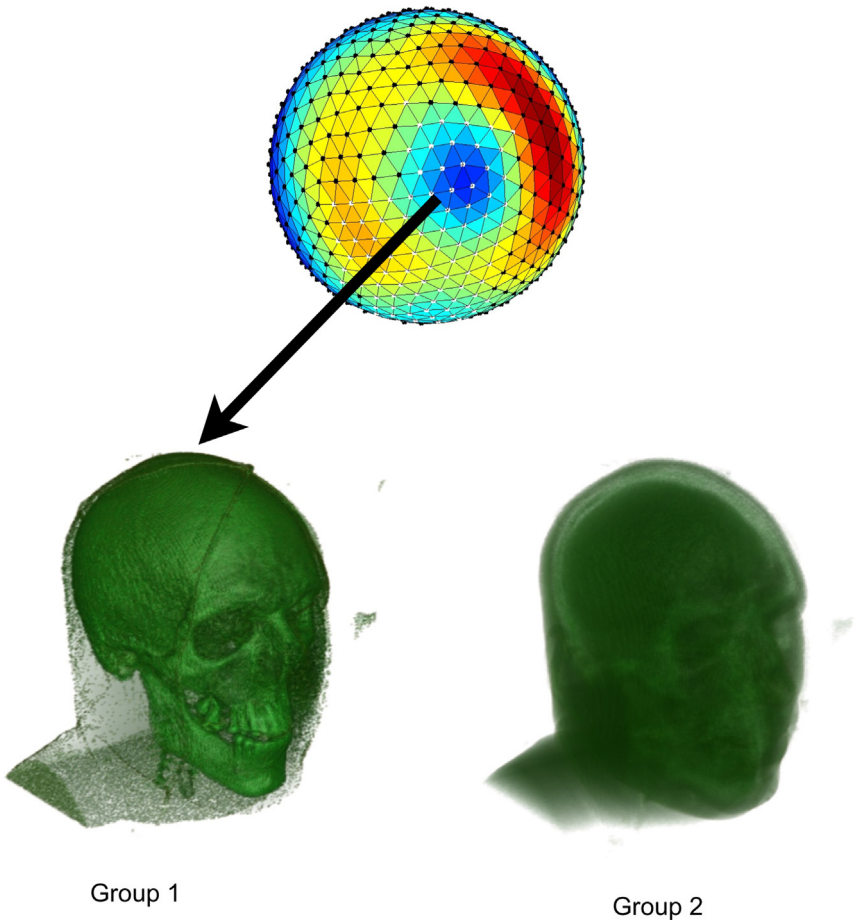

Fig. 10. Different user-selected voxel groups and corresponding renderings of the Visual Male Head dataset. White and blue dots refer to Group 1 and Group 2 (pointed out by the arrows), respectively. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

Fig. 11. Rendering of the Visual Male Head dataset (all groups together).

Figs. 12 and 13 show the results of the other datasets. These renderings were obtained with two user-defined groups and the following hue settings:

- *Carp*: L-type hue template starting at 144°.
- *Engine*: V-type hue template starting at 90°.
- *Lobster*: T-type hue template starting at 18°.

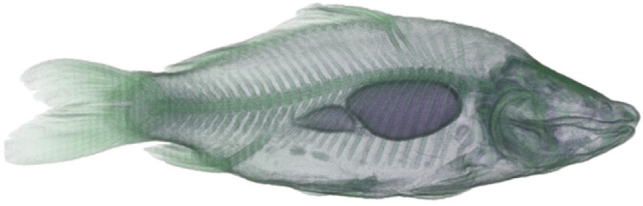

Fig. 12. Rendering of the Carp dataset with two groups.

Comparing these results with the renders in Fig. 4 (obtained without any grouping or automatic color assignment), we can clearly see that the flexibility of user-defined voxel grouping on the SSOM lattice and the robustness of our automatic color and opacity assignment result in the exposure of detailed structures in all the datasets.

The size of each dataset, training times and number of training iterations used are listed in Table 1. Note that the training of a SSOM has to be done only once for each dataset and does not effect the rendering process of our proposed method, which is required to be real-time. Also, as shown before, due to the high number of voxels with similar features, 1–2 iterations of training is good enough for stabilization of the map. Such a low number of iterations is reasonable here, since the perfect convergence of map is not very critical, as long as we can have a color-coded SSOM where different cluster regions and the borders between them are visually distinguishable.

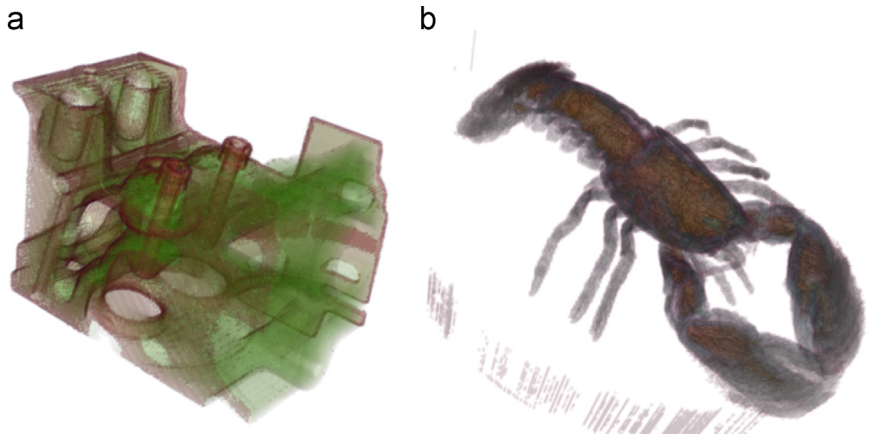

Fig. 13. (a) Rendering of the Engine dataset with two groups. (b)Rendering of the Lobster dataset with two groups.

Table 1
The size of the volume datasets, number of iterations for training and the required SSOM training times (in *seconds*).

Dataset name	Size	Iterations	Training time
Foot	256 × 256 × 256	1	442.5
Visual Male Head	128 × 256 × 256	2	1524.6
Carp	256 × 256 × 512	2	2395.9
Engine	256 × 256 × 256	1	369.7
Lobster	301 × 324 × 56	1	111.11

To further strengthen the usability of our system, we conducted a user survey, where we compared our proposed system with an off-the-shelf traditional TF editor [41]. A point to note here is that to our knowledge, there is no existing method to quantitatively compare volume rendering techniques, since the final interpretation is user dependent. Some measures have been proposed recently to tackle this problem [44]. Even those measurements had to undergo extensive user studies [44] to justify their applicability. The problem with evaluation of volume rendering systems lies in the fact that there are too many different parameters from system to system to come up with a fair and viable comparison platform. Most of the proposed methods make use of their own unique interfaces to make the user interaction easier and more efficient. The TF generation process also varies significantly. As a result, we have resorted to the commonly accepted user satisfaction and efficiency as comparison tools [45,46].

For our user survey, three users were carefully chosen to represent varying level of knowledge and familiarity with visualization tools and techniques (User 1 being the most familiar while User 3 the least). There were two phases in the survey:

- *Training*: In this phase, the users were familiarized with the interface of the systems. This phase was continued till the user was comfortable with the various options (e.g. selecting voxels, defining voxel groups in our proposed system and selecting points from histogram in the traditional TF editor).
- *Evaluation*: The users were provided with the raw visualization of the Foot dataset without any voxel grouping (Fig. 4(a)) and a reference image with three voxel groups (Fig. 9(b)). The task was to play with the different options in the system and generate a similar rendering as the reference image from the raw visualization. In our proposed system, the color and opacity values are automatically generated through color harmonization. For the traditional TF editor, the user was free to chose arbitrary color and opacity values. Since the reference image was generated from our system, to be fair to the traditional method, matching the color and opacity was not a requirement, as long as all three voxel groups were distinguishable to a satisfactory level for the user.

Table 2
Times recorded for user training and interaction (in *seconds*) and the ratings (on a scale of 5) by the users for interaction and output quality (proposed method in bold).

User	System	Training time	Interaction time	Interaction rating	Output rating
User 1	Proposed	**110**	**130**	**4.5**	**4**
	Traditional	170	250	3.5	2.5
User 2	Proposed	**100**	**130**	**5**	**4.5**
	Traditional	210	340	3.5	2.5
User 3	Proposed	**170**	**220**	**4**	**5**
	Traditional	580	610	3	2.5
Average	Proposed	**126.67**	**160**	**4.5**	**4.5**
	Traditional	320	400	3.33	2.5

During these two phases, the time required was recorded for each user (Table 2). After the evaluation, the users were asked to rate each system on a Likert scale [47] of 5, with 5 being the best (completely satisfied) and 1 being the worst (not satisfied). There were two categories for rating, the interaction with the system and the quality of the output.

Table 2 summarizes the results from this user survey. As we can see, our proposed method performs significantly better in all aspects. For the training phase, we see that all the three users needed more time to adapt to the traditional TF editor than to our proposed system. The time difference is especially significant in case of User 3, who happens to be the least experienced with visualization systems. To interact with the traditional TF editor, you need some knowledge of histogram and color/opacity mapping. Since this user was not familiar with these concepts, it took him or her more time to adapt. However, our proposed system is simple and intuitive. No low level parameter is exposed to the user. As a result, the training for our system was faster. Even with experienced users like User 1 or User 2, we see that the training of our proposed system was faster.

In case of interaction time, we see that our proposed system could generate satisfactory output very quickly, while with the traditional editor it took more time. With the traditional editor, on top of histogram manipulation, the user had to assign the color and opacity values manually. As a result, it took longer to generate a satisfactory output. The same improvements are noticeable in case of user ratings. We see that our system got an average rating of 4.5 for both interaction and output satisfaction, while the traditional system only got 3.33 and 2.5 respectively. The traditional TF was especially unsatisfactory in terms of output quality. This is

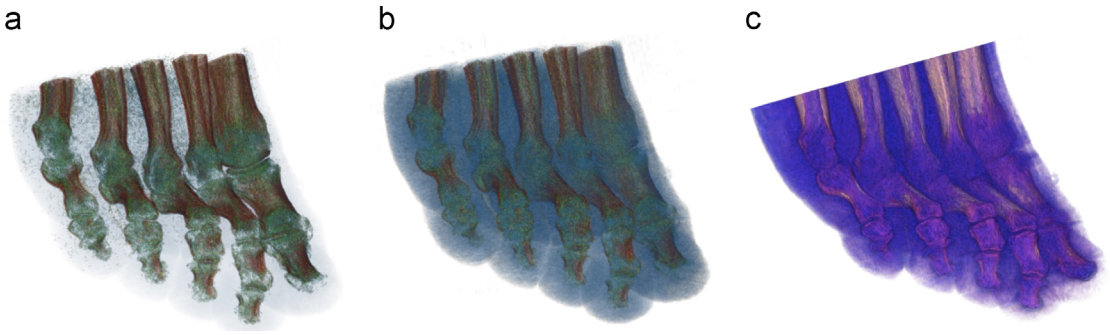

Fig. 14. Rendering of the Foot dataset from different TF editors. (a) Proposed (two groups). (b) Proposed (three groups). (c) Traditional.

expected, as the traditional editor only makes use of intensity histograms. Our proposed system uses high-dimensional features. As a result, the voxel groups can be identified easily and efficiently. In fact, the users were not able to differentiate the three voxel groups in the Foot dataset accurately with the traditional TF editor. As we can see in Fig. 14(c), the voxel group near the bone joints is not distinguishable. But with our proposed system, the group is easily identifiable (Fig. 14(a)). Our proposed system is also flexible in terms of generating different visualizations, as the voxel group visibility can be turned on or off with just one click. No such operation is possible with the traditional TF system.

5. Conclusion

In this paper, we have proposed a new intuitive way of direct volume rendering with the help of spherical self-organizing maps and color harmonization. The user interacts with the SSOM lattice to find interesting regions and group SSOM nodes together. The TF is generated using color harmonization and the user's perception, which results in an aesthetically pleasing rendering where the detailed structures inside a volume are easily distinguishable. Real-time rendering of the generated TF on the volume dataset provides instant feedback to the user. The proposed system is intuitive to interact with and robust in nature, since any feature can be used with the SSOM lattice. Experimental results on the five volume datasets verify the feasibility of our proposed approach. In future, we plan to extend this system to learn from user interaction and generate knowledge-assisted volume rendering accordingly.

Appendix A. Supplementary material

Supplementary data associated with this article can be found in the online version of http://dx.doi.org/10.1016/j.neucom.2013.09.064.

References

[1] G. Kindlmann, J. Durkin, Semi-automatic generation of transfer functions for direct volume rendering, in: Proceedings of the 1998 IEEE Symposium on Volume visualization, ACM, Research Triangle Park, North Carolina, USA, 1998, pp. 79–86.
[2] B. Nguyen, W.-L. Tay, C.-K. Chui, S.-H. Ong, A clustering-based system to automate transfer function design for medical image visualization, Vis. Comput. 28 (2012) 181–191.
[3] Y. Wang, W. Chen, J. Zhang, T. Dong, G. Shan, X. Chi, Efficient volume exploration using the Gaussian mixture model, IEEE Trans. Vis. Comput. Graph. 17 (2011) 1560–1573.
[4] M. Selver, M. Alper, C. Guzeli, Semiautomatic transfer function initialization for abdominal visualization using self-generating hierarchical radial basis function networks, IEEE Trans. Vis. Comput. Graph. 15 (2009) 395–409.
[5] R. Maciejewski, I. Wu, W. Chen, D. Ebert, Structuring feature space: a non-parametric method for volumetric transfer function generation, IEEE Trans. Vis. Comput. Graph. 15 (2009) 1473–1480.
[6] S. Roettger, M. Bauer, M. Stamminger, Spatialized transfer functions, in: IEEE/Eurographics Symposium on Visualization, IEEE Press, Konstanz, Germany, 2005, pp. 271–278.
[7] T. Kohonen (Ed.), Self-Organizing Maps, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997.
[8] S. Haykin, Neural Networks: A Comprehensive Foundation, MacMillan, New York, 1999.
[9] Y. Liu, C. Lisle, J. Collins, Quick2insight: a user-friendly framework for interactive rendering of biological image volumes, in: IEEE Symposium on Biological Data Visualization, IEEE Press, Providence, RI, USA, 2011, pp. 1–8.
[10] D. Cohen-Or, O. Sorkine, R. Gal, T. Leyvand, Y.-Q. Xu, Color harmonization, ACM Trans. Graph. 25 (2006) 624–630.
[11] J. Zhou, M. Takatsuka, Automatic transfer function generation using contour tree controlled residue flow model and color harmonics, IEEE Trans. Vis. Comput. Graph. 15 (2009) 1481–1488.
[12] M. Tokumaru, N. Muranaka, S. Imanishi, Color design support system considering color harmony, in: IEEE International Conference on Fuzzy Systems, IEEE Press, Honolulu, HI, USA, 2002, pp. 378–383.
[13] A. Sangole, A. Leontitsis, Spherical self-organizing feature map: an introductory review, Int. J. Bifurc. Chaos 16 (2006) 3195–3206.
[14] A. Sangole, Data-driven modeling using spherical self-organizing maps (Ph.D. thesis), Department of Mechanical and Materials Engineering, The University of Western Ontario, 2002.
[15] H. Tokutaka, M. Ohkita, Y. Hai, K. Fujimura, M. Oyabu, Classification using topologically preserving spherical self-organizing maps, in: Workshop on Self-Organizing Maps Lecture Notes on Computer Science, vol. 6731, Springer-Verlag, Espoo, Finland, 2011, pp. 308–317.
[16] N. Khan, M. Kyan, L. Guan, Intuitive volume exploration through spherical self-organizing map, in: Ninth Workshop on Self-Organizing Maps, Springer-Verlag, Santiago, Chile, 2012, pp. 75–84.
[17] M. Levoy, Display of surfaces from volume data, IEEE Comput. Graph. Appl. 8 (1988) 29–37.
[18] J. Kniss, G. Kindlmann, C. Hansen, Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets, in: IEEE Symposium on Volume Visualization, IEEE Press, San Diego, California, USA, 2001, pp. 255–262.
[19] P. Sereda, A. Bartroli, I. Serlie, F. Gerritsen, Visualization of boundaries in volumetric data sets using LH histograms, IEEE Trans. Vis. Comput. Graph. 12 (2006) 208–218.
[20] C. Bajaj, V. Pascucci, D. Schikore, The contour spectrum, in: Eighth Conference on Visualization, IEEE Press, Phoenix, AZ, USA, 1997, pp. 167-ff.
[21] G. Kindlmann, R.W.T. Ross, T. Tasdizen, T. Moller, Curvature-based transfer functions for direct volume rendering: methods and applications, IEEE Press, Seattle, Washington, USA, 2003, pp. 513–520.
[22] C. Correa, K. Ma, Size-based transfer functions: a new volume exploration technique, IEEE Trans. Vis. Comput. Graph. 14 (2008) 1380–1387.
[23] M. Hadwiger, L. Fritz, C. Rezk-Salama, T. Hollt, G.G.T. Pabel, Interactive volume exploration for feature detection and quantification in industrial ct data, IEEE Trans. Vis. Comput. Graph. 14 (2008) 1507–1514.
[24] S. Wesarg, M. Kirschner, M. Khan, 2d histogram based volume visualization: combining intensity and size of anatomical structures, IEEE Trans. Vis. Comput. Graph. 5 (2010) 655–666.
[25] C. Lundstrom, P. Ljung, A. Ynnerman, Local histograms for design of transfer functions in direct volume rendering, IEEE Trans. Vis. Comput. Graph. 12 (2006) 1570–1579.
[26] C. Lundstrom, A. Ynnerman, P. Ljung, A. Persson, H. Knutsson, The α-histogram: using spatial coherence to enhance histograms and transfer function design, in: IEEE Symposium on Visualization, IEEE Press, Baltimore, Maryland, USA, 2006, pp. 227–234.
[27] A. Tappenbeck, B. Preim, V. Dicken, Distance-based transfer function design: specification methods and applications, in: SimVis, IEEE Press, Magdeburg, Germany, 2006, pp. 259–274.

[28] F. Pinto, C.M.D.S. Freitas, Design of multi-dimensional transfer functions using dimensional reduction, in: Eurographics Symposium on Visualization, IEEE Press, Sacramento, California, USA, 2007, pp. 131–138.
[29] S.S. Fang, T.B. Tom, M. Tuceryan, Image-based transfer function design for data exploration in volume visualization, in: Conference on Visualization, VIS '98, IEEE Press, Research Triangle Park, North Carolina, USA, 1998, pp. 319–326.
[30] J. Marks, B. Andalman, P. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims, S. Shieber, Design galleries: a general approach to setting parameters for computer graphics and animation, in: Annual Conference on Computer Graphics and Interactive Techniques, ACM, Los Angeles, California, USA, 1997, pp. 389–400.
[31] H. Guo, H. Xiao, X. Yuan, Multi-dimensional transfer function design based on flexible dimension projection embedded in parallel coordinates, in: IEEE Pacific Visualization Symposium, IEEE Press, Hong Kong, 2011, pp. 19–26.
[32] F.-Y. Tzeng, Intelligent system-assisted user interfaces for volume visualization (Ph.D. thesis), University of California, 2006.
[33] B. Liu, B. Wünsche, T. Ropinski, Visualization by example - a constructive visual component-based interface for direct volume rendering, Scite Press, Angers, France, 2010, pp. 254–259.
[34] P. Sereda, A. Vilanova, F. Gerritsen, #Automating transfer function design for volume rendering using hierarchical clustering of material boundaries, in: IEEE Symposium on Visualization, IEEE Press, Baltimore, Maryland, USA, 2006, pp. 243–250.
[35] F. Zhou, Y. Zhao, K. Ma, Parallel mean shift for interactive volume segmentation, in: International Conference on Machine Learning in Medical Imaging, IEEE Press, Beijing, China, 2010, pp. 67–75.
[36] H. Ritter, Self-organizing Maps in Non-Euclidean Spaces, Springer, Germany.
[37] Y. Wu, M. Takatsuka, The geodesic self-organizing map and its error analysis, in: Australasian conference on Computer Science, Australian Computer Society, Inc., Newcastle, NSW, Australia, 2005, pp. 343–351.
[38] J. Hladuvka, A. Konig, E. Groller, Exploiting eigenvalues of the hessian matrix for volume decimation, in: International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, Vaclav Skala, 2001, pp. 124–129.
[39] A. Krishnamurthy, S. Ahalt, D. Melton, P. Chen, Neural networks for vector quantization of speech and images, IEEE J. Sel. Areas Commun. 8 (1990) 1449–1457.
[40] J. Itten, The Art of Color: The Subjective Experience and Objective Rationale of Color, Van Nostrand Reinhold Company, New York, New York, United States, 1969.
[41] W. Schroeder, K. Martin, B. Lorensen (Eds.), The Visualization Toolkit, Kitware, New York, NY, USA, 2006.
[42] S. Roettger, The Volume Library, 2012. Ohm Hochschule Nurnberg, Nurnberg, Germany.
[43] W.K. Pratt (Ed.), Digital Image Processing, John Wiley and Sons, New York, NY, USA, 2007.
[44] Y. Wu, H. Qu, K.-K. Chung, M.-Y. Chan, Quantitative effectiveness measures for direct volume rendered images, in: IEEE Pacific Visualization Symposium, IEEE Press, Sydney, NSW, Australia, 2007, pp. 1–8.
[45] J. Giesen, K. Mueller, E. Schuberth, L. Wang, P. Zolliker, Conjoint analysis to measure the perceived quality in volume rendering, IEEE Trans. Vis. Comput. Graph. 13 (2007) 1664–1671.
[46] V. Walimbe, V. Zagrodsky, S. Raja, W.A. Jaber, F.P. DiFilippo, M.J. Garcia, R.C. Brunken, J.D. Thomas, R. Shekhar, Mutual information-based multimodality registration of cardiac ultrasound and spect images: a preliminary investigation, Int. J. Cardiovasc. Imag. 19 (2003) 483–494.
[47] R. Likert, A technique for the measurement of attitudes, Arch. Psychol. 22 (1932) 1–55.

Naimul Mefraz Khan is a Ph.D. student at Ryerson Multimedia Research Laboratory. He completed his B.Sc. in Computer Science & Engineering from Bangladesh University of Engineering & Technology (BUET) and his M.Sc. in Computer Science from the University of Windsor. His areas of expertise include machine learning, computer graphics and image processing.

Matthew Kyan received the B.Sc. degree in Computer Science and the B.Eng. and Ph.D. degrees in Electrical Engineering from the University of Sydney, NSW, Australia, in 1997, 1998, and 2007, respectively. He is currently an Assistant Professor at Ryerson University, Toronto, Canada, where he was instrumental in establishing the Center for Interactive Multimedia Information Mining (CIM2). With more than 30 internationally refereed publications, his research interests include evolutionary computation, pattern recognition, visual attention, multimedia summarization, knowledge-assisted visualization, mixed and shared reality, biometrics, and bioinformatics. His research addresses unsupervised learning through dynamic self-organization, applied to multimedia signal processing, and information mining. He received the Inaugural Siemens National Prize for Innovation, Australia, in 1999, for his work with 3D confocal image analysis. He is a member of the IEEE Signal Processing, Computational Intelligence, and Engineering in Medicine and Biology Societies. He is currently serving as Treasurer for IEEE EMBS, Toronto Chapter.

Ling Guan received his bachelor's degree from Tianjin University, China, master's degree from University of Waterloo, Canada, and Ph.D. degree from University of British Columbia, Canada. From 1992 to 2001, he was on the Faculty of Engineering, University of Sydney, Australia. Since 2001, he has been a Tier I Canada Research Chair in Multimedia and Computer Technology, and a Professor of Electrical and Computer Engineering at Ryerson University, Toronto, Canada. He is the Founder and Director of Ryerson Multimedia Research Lab (an icon of Canadian Excellence in multimedia research) and Centre for Interactive Multimedia Information Mining. He has been working on image, video and multimedia signal processing, human–computer interaction, pattern recognition and machine intelligence, and authored/co-authored more than 350 scientific publications. He has served on half a dozen editorial boards of IEEE Transactions and Magazines, chaired the 2006 IEEE International Conference on Multimedia and Expo in Toronto and co-chaired the 2008 ACM International Conference on Image and Video Retrieval in Niagara Falls. He played the leading role in the inauguration of IEEE Pacific-Rim Conference on Multimedia in 2000 and served as the Founding General Chair. He is a Fellow of the IEEE, the Canadian Academy of Engineering and the Engineering Institute of Canada. He is an IEEE Circuits and System Society Distinguished Lecturer and a recipient of the 2005 IEEE Transactions on Circuits and Systems for Video Technology Best Paper Award.