Intuitive Volume Exploration through Spherical Self-Organizing Map

Naimul Mefraz Khan, Matthew Kyan, and Ling Guan

Ryerson University, Toronto, ON
{n77khan,mkyan,lguan}@ee.ryerson.ca

Abstract. Direct Volume Rendering is one of the most popular volume exploration methods, where the data values are mapped to optical properties through a Transfer Function (TF). However, designing an appropriate TF is a complex task for the end user, who may not be an expert in visualization techniques. The Self-Organizing Map (SOM) is a perfect tool to hide irrelevant TF parameters and, through unsupervised clustering, present a visual form of the topological relations among the clusters. This paper introduces a novel volume exploration technique which utilizes the cluster visualization ability of SOM to present a simple intuitive interface to the user for generating suitable TFs. Rather than manipulating TF or cluster parameters, the user interacts with the spherical lattice of the SOM to discover interesting regions in the volume quickly and intuitively. The GPU implementation provides real-time volume rendering and fast interaction. Experimental results on several 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. 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. The user can also control some low-level properties like number of clusters, cluster variance etc. Most of the recently proposed DVR methodologies [1,6,16,13] 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. Also, these kind of widgets try to represent the feature space directly, putting a restriction on the dimensionality of the feature space. Some methods use alternative ways to represent the higher-dimensional feature space through manageable widgets. For instance, in [10], spatial information is encoded in the color values while opacity is derived through intensity and gradient. But these kind 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 wide range of applications in different fields, and one set of features useful for a specific application might be completely irrelevant in another. Hence, there is a need to make the method independent so that any feature irrespective of its dimensionality can be represented to the user in a visual form while maintaining the topological relationship between various data distributions. This is exactly what a Self-Organizing Map (SOM) [8] can do. SOM preserves the input data topology and helps to generate a lower dimensional visualization of the clusters. The SOM structure is particularly of interest for DVR because of it's visualization capability.

This paper proposes such a DVR system where the feature space is represented to the user with the help of SOM. Our proposed system has the following advantages over existing DVR techniques:

- 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 proposed model is independent of the dimensionality of the feature space. Any feature irrespective of its dimension or complexity can be used with the model, which makes it very robust.

We use the Spherical SOM structure (SSOM) [11,14] because of it's relatively less restrictive structure (explained later) . The GPU implementation of our method provides fast interaction with the SOM and real-time volume rendering.

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

As TF specification is a huge research area itself, only recent and relevant works will be briefly discussed here. Traditionally, TF were only one-dimensional, where the color and opacity is derived from intensity value only. The work of Kindlmann and Durkin first popularized multi-dimensional TF, where intensity and gradient magnitude value were used to generate a 2D histogram to emphasize boundaries between different materials in a volume. Since then, many methods have been proposed to simplify the interaction with a multi-dimensional TF. A semi-automatic TF generation based on Radial

Basis Function (RBF) networks is presented in [13]. A nonparametric density estimation technique is introduced in [5]. A Gaussian Mixture Model-based clustering technique is presented in [16], where the Gaussians can be mapped to a set of elliptical transfer functions. A mixed model based on mean-shift and hierarchical clustering on the Low-High (LH) values of a volume is described in [1]. Since including only the intensity and gradient value results in local features, Roettger et al. [10] propose transfer functions that consider spatial information for clustering on 2D histograms. An intelligent user interface has been introduced in [15], 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 [4], where the user can combine simpler pre-defined TFs to generate complex ones.

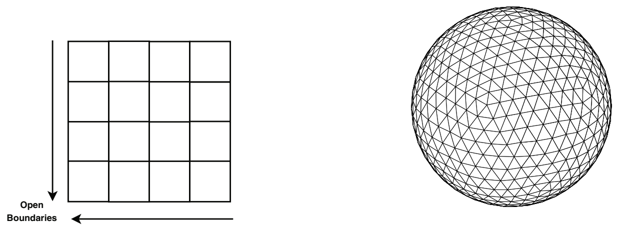
Despite all these efforts to make TF specification a simple and intuitive task, most of these methods still rely on some form of user control (e.g. number of clusters, variance of clusters, merging and splitting of clusters) in the feature space. An expert from medical or architectural background might not be familiar with these specifications. 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 (SOM).

Our method draws some inspiration from the method proposed in [7]. However, the SOM visualization and volume exploration in our proposed method is 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 [7] mainly for dimensionality reduction. The Gaussian TF generation in [7] 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. Lastly, in [7], 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 [11] (details in Section 3), which can result in faster training times. We also use the Spherical Self-Organizing Map for it's 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 [8] 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, due to the easy visualization property of SOM, it is suitable for presenting volumetric data to the user for TF design.

The traditional SOM however has a rectangular 2D grid structure, which has a restricted boundary. This is because the boundaries are open (Figure 1), and the nodes on the boundary do not have the same number of neighbors as the inner nodes. Nodes on the opposite sides of the boundary are not topologically close in the SOM space. The same is true for even a 3D cubic structure [11]. In some cases, a wrap-around is introduced on the 2D structure to eliminate this boundary condition. However, this introduces other problems such as folding and twisting [11]. The ideal structure for SOM will be a lattice that minimizes topological discontinuity. Hence, we use the Spherical SOM (SSOM) [11] for our method. The SSOM structure is created by repeatedly subdividing an Icosahedron. This provides a spherical structure with symmetric node distribution.



(a) Open boundaries of 2D SOM (b) Closed structure of Spherical SOM

Fig. 1. Depiction of closed structure of spherical SOM compared to the traditional 2D SOM

As seen in Figure 1, the SSOM does not have the restricted boundary problem. The SSOM is also of particular interest because of it's 3D structure, which can be easier to navigate.

3 Proposed Method

In this section, we describe the details of our proposed system based on the 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 is described and 3) TF representation and exploration, where the simple and efficient manner in which a user can explore different Transfer Functions through the visual representation of an SSOM is explained.

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 [10]. 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 histogram-based approaches, the spatial information is used for color generation only [10]. However, as described before, our proposed model is independent of feature dimensionality due to the use of SSOM lattice. No matter how high the dimension of our feature set is, we can map it to a Spherical SOM and represent the clustering visually. 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 and it's 3D gradient magnitude. Our 5D feature set consist of the following features:

- The X,Y and Z coordinates,
- the intensity value and
- the 3D gradient magnitude. The 3D gradient magnitude is defined by $G = \sqrt{G_x^2 + G_y^2 + G_z^2}$, where G_x,G_y and G_z are the gradient values along X, Y and Z direction, respectively.

All the features are scaled to fall between the value 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 $\{0.5, 0.5, 0.5, 1, 2\}$ for the aforementioned features.

3.2 SSOM Training

The training phase of the SSOM is similar to the classical SOM [8]. Let, the input space of N voxels be represented by $\mathscr{X} = \{x_i\}_{i=1}^{N}$. Let, the SSOM be represented by M nodes ($M << N$). Each node in the SSOM lattice has a corresponding weight vector w. All these weight vectors together represent our SSOM space $\mathscr{W} = \{w_i\}_{i=1}^{M}$. 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 [11].

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 usually used as distance measure. The update step then takes place, where the weight vector of the BMU and it's neighboring nodes (Θ_{BMU}^r) are updated in a way so 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 represents the underlying distribution of the input data (the voxel features in this case). The 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. in [8], random initialization will take more time to converge. We have followed the initialization method stated in [8] 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. A count vector $\mathscr{C} = \{c_i\}_{i=1}^{M}$ [11] 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 (explained below) to prevent cluster under-utilization. This is especially necessary in our case because in a volume, typically there is almost $70\% - 80\%$ homogeneous regions. Without a count-dependent control, homogenous regions will take over the whole map and important regions (boundaries) will not get enough map space to be noticed.

- **Training:** For each input voxel x, do the following:
  - **BMU Selection:** Calculate the Euclidean distance of the voxel feature vector x with all the nodes as follows:

$$e_i = (c_i + 1)\|x - w_i\|, \quad i = 1, \dots, M. \tag{1}$$

    BMU is the node for which this distance is the smallest.
  - **Weight Update:** Update the weights for the BMU and it's neighboring nodes (defined by Θ^r_{BMU}) as follows:

$$w = w + b(t) * h(s,r) * \|x - w\|, \tag{2}$$

$$c_w = c_w + h(s,r), \tag{3}$$

    where $w \in \Theta^r_{BMU}$, $b(t) = \alpha e^{-\frac{t}{T}}$ and $h(s,r) = e^{-\frac{r^2}{s*R}}$.

    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 epoch number $t = 1, 2, \dots, T$ 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 it's weight will be affected.

    As discussed before, the count-dependent parameter c_w is increased here to prevent cluster under-utilization. Observing Equation (3) and Equation (1), we 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 [3].
- Repeat the training steps for a pre-defined number of epochs (T).

The main control parameters in SSOM training are the learning rate α, the number of epochs 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 $256X256X256$, there are over 16 million voxels). As we have found experimentally, for such high number of voxels, the SOM typically converges within $2 - 3$ epochs. 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 interaction options to select interesting regions in the SSOM, 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 [11], which visualizes 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. In the proposed system, the user can select or de-select any region of the spherical lattice. The selection is provided in the form of a rubber-band tool, where the user can drag across the surface of the sphere to select one or multiple nodes. The de-selection is performed in a similar way.

The last step of our system is to map the SSOM to a TF. The TF is essentially an RGBA texture with corresponding entry for each voxel. The RGB corresponds to the voxel color, while the alpha component defines the voxel opacity. To speed up the rendering process, we keep the RGB channels fixed throughout a session. The alpha value is the most crucial component, since the opacity determines the visibility of a voxel. In our proposed model, we 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. In this way, the boundaries between the materials will be colored differently.

The opacity of a voxel depends on the user selection. For voxels corresponding to the selected region $\mathscr{S}$ of the map, the opacity is calculated through following equation:

$$I(x) = 1 - \frac{\|w_{BMU_x} - x\|}{max_{y \in \mathscr{S}}(\|w_{BMU_y} - y\|)}, \tag{4}$$

where x represents the voxel for which the opacity is being calculated, and y represents all other voxels under the selected region $\mathscr{S}$. This essentially means that the closer the voxel will be to the weight vector of its winning node, the more opaque (less transparent) it will be. This assigns intuitive opacity values to the voxels corresponding to the selected SSOM nodes. This value is then scaled to be in a higher opaque region (0.6 – 1.0 in our experiments). The voxels corresponding to the unselected regions are assigned a very low but nonzero opacity value (0.01) so that the user still has some context while viewing the rendering of the selected region.

4 Experimental Results

To show the effectiveness of our system, we present some volume rendering results on popular datasets. We used three volume datasets [9], CT scans of a Foot, an Engine and a Piggy Bank. The size of the datasets are listed in Table 1. Generally, CT scan 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 [2] was used to separate the object from these regions first. These separated voxels were then fed to the SSOM training algorithm.

As stated before, due to the high number of voxels (in the range of millions), $2-3$ epochs of training is good enough for a visualization of the cluster. Such low number of epochs is reasonable here, since the 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. The training times required for the datasets are listed in Table 1. Please 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.

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

Dataset Name	Size	Training Time
Foot	256X256X256	342.3
Engine	256X256X256	308.3
Piggy Bank	512X512X134	1341.4

After the training, we present a visual form of the spherical lattice to the user (details in Section 3.3). The user can then select or de-select regions of interest. The voxels of the selected regions were assigned opacity values using Equation (4). The RGBA texture according to the generated TF is then rendered on the GPU in real time. We have used the Visualization Toolkit [12] for our GPU rendering purposes.

Figure 2 shows some results obtained from our experiments. The top row (Figure 2.(a), (d), (g)) shows the rendering results when the full SSOM is selected. As we can see, although the surfaces of the volumes are visualized clearly, not much useful structural information can be gathered from these renderings. Figure 2.(b), (e), (h) shows the SSOMs corresponding to the three datasets with some parts of the maps selected by the user (the selected nodes are highlighted by white spots). Figure 2.(c), (f), (i) shows the corresponding renderings from the selected nodes. As we can see, for all three volumes, the important structures can be highlighted easily. The inner bones are visible in the Foot dataset (Figure 2.(c)). Similarly, the tubes of the Engine are visible and the coins inside the Piggy Bank are visible (Figure 2.(f), (i)). This clearly shows the effectiveness and efficiency of our method. Another interesting observation here is the nature of the selected regions on the map. As we can see, the selected regions contain

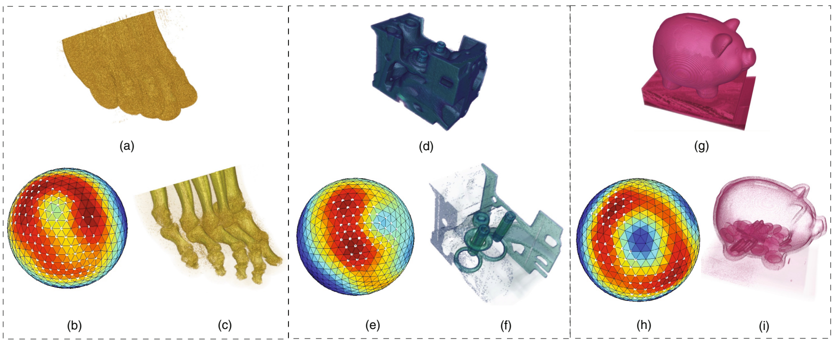


Fig. 2. Rendering results and corresponding SSOMs for the three datasets, (a)-(c): Foot; (d)-(f): Engine and (g)-(i): Piggy Bank

overlapping cluster regions in some cases. This is where the power of SSOM clustering is apparent. Since the clustering is not strict and the user has the freedom to select whatever region he or she wants, the whole process is very flexible. Also, due to the spherical structure of the map, it is easy to generate customized rendering depending on the users' need very easily. All the user has to do is select the appropriate regions on the map.

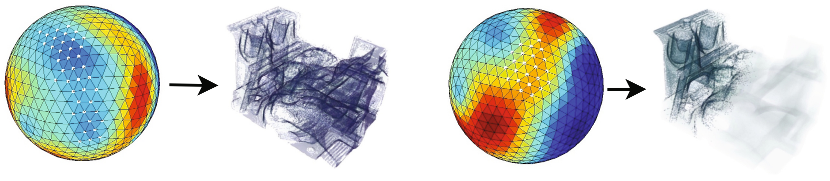


Fig. 3. Additional renderings corresponding to selected regions on the SSOM for the Engine

Figure 3 shows additional renderings corresponding to different selected regions on the SSOM. As we can see, the first region corresponds to a silhoutte-like rendering, while the second rendering focuses on the rear part of the engine. Since the selected region can be as small as only a single node, the system is very robust and adaptable.

5 Conclusion

In this paper, we have proposed a new intuitive way of direct volume rendering with the help of Spherical Self-Organizing Maps. The user interacts with the SSOM lattice to find interesting regions and generate suitable TF. 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 irrespective of its complexity. Experimental results on some popular 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.

References

1. Kindlmann, G., Durkin, J.W.: Semi-automatic generation of transfer functions for direct volume rendering. In: Proceedings of the 1998 IEEE Symposium on Volume Visualization, pp. 79–86. ACM, New York (1998)
2. Kohonen, T.: Self-organizing maps. Springer-Verlag New York, Inc., Secaucus (1997)
3. Krishnamurthy, A., Ahalt, S., Melton, D., Chen, P.: Neural networks for vector quantization of speech and images. IEEE Journal on Selected Areas in Communications 8, 1449–1457 (1990)
4. Liu, B., Wünsche, B., Ropinski, T.: Visualization by example - a constructive visual component-based interface for direct volume rendering. In: Proceedings of the International Conference on Computer Graphics Theory and Applications, pp. 254–259 (2010)
5. Maciejewski, R., Wu, I., Chen, W., Ebert, D.: Structuring feature space: A non-parametric method for volumetric transfer function generation. IEEE Transactions on Visualization and Computer Graphics 15, 1473–1480 (2009)
6. Nguyen, B., Tay, W., Chui, C., Ong, S.: A clustering-based system to automate transfer function design for medical image visualization. The Visual Computer 28, 181–191 (2012)
7. Pinto, F., Freitas, C.M.D.S.: Design of multi-dimensional transfer functions using dimensional reduction. In: Proceedings of the Eurographics Symposium on Visualization, pp. 131–138 (2007)
8. Pratt, W.K. (ed.): Digital Image Processing. John Wiley and Sons, New York (2007)
9. Roettger, S.: The volume library. Ohm Hochschule Nurnberg (2012), `http://www9.informatik.uni-erlangen.de/External/vollib/`
10. Roettger, S., Bauer, M., Stamminger, M.: Spatialized transfer functions. In: Proceedings of the IEEE/Eurographics Symposium on Visualization, pp. 271–278 (2005)
11. Sangole, A., Leontitsis, A.: Spherical self-organizing feature map: An introductory review, pp. 3195–3206 (2006)
12. Schroeder, W., Martin, K., Lorensen, B. (eds.): The Visualization Toolkit. Kitware, New York (2006)
13. Selver, M., Alper, M., Guzeli, C.: Semiautomatic transfer function initialization for abdominal visualization using self-generating hierarchical radial basis function networks. IEEE Transactions on Visualization and Computer Graphics 15, 395–409 (2009)
14. Tokutaka, H., Ohkita, M., Hai, Y., Fujimura, K., Oyabu, M.: Classification Using Topologically Preserving Spherical Self-Organizing Maps. In: Laaksonen, J., Honkela, T. (eds.) WSOM 2011. LNCS, vol. 6731, pp. 308–317. Springer, Heidelberg (2011)
15. Tzeng, F.Y.: Intelligent system-assisted user interfaces for volume visualization. Ph.D. thesis, University of California, Davis, CA, USA (2006)
16. Wang, Y., Chen, W., Zhang, J., Dong, T., Shan, G., Chi, X.: Efficient volume exploration using the gaussian mixture model. IEEE Transactions on Visualization and Computer Graphics 17, 1560–1573 (2011)