Eurographics/ IEEE-VGTC Symposium on Visualization (2007)
Ken Museth, Torsten Möller, and Anders Ynnerman (Editors)

Design of Multi-dimensional Transfer Functions Using Dimensional Reduction

Francisco de Moura Pinto and Carla M. D. S. Freitas

Instituto de Informática, UFRGS, Brazil

Abstract

Direct volume rendering techniques allow visualization of volume data without extracting intermediate geometry. The mapping from voxel attributes to optical properties is performed by transfer functions which, consequently, play a crucial role in building informative images from the data. One-dimensional transfer functions, which are based only on a scalar value per voxel, often do not provide proper visualizations. On the other hand, multi-dimensional transfer functions can perform more sophisticated data classification, based on vectorial voxel signatures. The transfer function design is a non-trivial and unintuitive task, especially in the multi-dimensional case. In this paper we propose a multi-dimensional transfer function design technique that uses self-organizing maps to perform dimensional reduction. Our approach gives uniform treatment to volume data containing voxel signatures of arbitrary dimension, and allows the use of any type of voxel attribute as part of the voxel signatures.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation

1. Introduction

In direct volume rendering (DVR), transfer functions (TFs) are used for emphasizing regions of interest inside volumes. The most common type of transfer function is the one-dimensional TF, which assigns optical properties (usually color and opacity) to voxels based only on their scalar value. Notwithstanding, one-dimensional TFs have a very limited classification power because they can not make distinction between volume regions defined by scalar values within the same range. On the other hand, multi-dimensional transfer functions can perform better classification because they take into account not only the scalar value of a voxel [KKH02], but also other attributes such as gradient magnitude, directional second derivative, curvature [KWTM03] and statistical measures [TLM01]. This way, sets of attributes are interpreted as multi-dimensional voxel signatures.

Designing an appropriate transfer function, even a one-dimensional TF, is a difficult task and much attention has been given to this issue in the literature [PLB*01]. As its domain increases, the interaction with and the visualization of the transfer function become more difficult. Thus, specifying multi-dimensional TFs is a very difficult problem.

In this paper we present a dimensional reduction method based on self-organizing maps (SOMs) and radial basis functions (RBFs) to simplify the design of multi-dimensional (nD) transfer functions. In a first step, nD voxel signatures calculated from the volume data are used to create a two-dimensional map; then, map coordinates are defined for all voxels. The transfer function design is performed in the two-dimensional map space (see details in Section 3). We explore the use of two types of SOMs: Kohonen maps [Koh97] and spherical self-organizing maps [SK02]. Since we reduce the dimension of nD signatures, volume data of any dimension can be treated uniformly, and any type of voxel attribute can be part of the signature, allowing selecting those that provide meaningful visualizations. We also propose a simple and effective user interface based on dual domain interaction [KKH02] as part of a visualization framework that uses graphics hardware to provide interactive volume rendering.

This paper is organized as follows. Next section presents the closest related works, including a brief review of some concepts. Section 3 describes our approach in detail, while main implementation aspects are discussed in Section 4. Section 5 analyzes results we obtained with our method, while Section 6 draws some conclusions and points directions for future work.

2. Related Work

2.1. Multi-dimensional Transfer Function Design

The design of multi-dimensional transfer functions brings two main challenges: exploration of the TF domain and actual representation of the TF for rendering purpose. It is possible to explicitly define a multi-dimensional transfer function by interacting in its domain with proper tools. Kniss et al. [KKH02] proposed a volume rendering environment containing a set of direct manipulation widgets for volume inspection, visualization of data distribution and design of three-dimensional transfer functions, using dual domain interaction.

However, the difficulty of exploring the transfer function domain increases with its dimensionality; therefore some approaches for transfer function design provide interfaces based on interaction in a simplified space. Region growing techniques were used by Huang and Ma [HM03] to segment volume data from seed points specified by the user; voxel signatures of the segmented region were used to automatically design a transfer function. Tzeng and Ma [TM04] clusterized voxel signatures by similarity allowing the user to specify the desired classification by successively splitting and merging the clusters. The user sees the results by associating visual properties to each material class. The same authors [TLM05] implemented multi-dimensional classification functions using neural networks and support vector machines. These functions were learned from training sets selected through a slice painting interface. The user paints the voxels of interest with a specific color, and the undesired ones with a different color. This way they implemented binary classification of voxels. Šereda et al. [vBG06] used hierarchical clustering to group voxels according to their LH signatures [vBSG06]. The user navigates through the hierarchy searching for the branches corresponding to regions of interest. Takanashi et al. [TLMM02] used independent component analysis (ICA) of multi-dimensional voxel signatures in order to represent them in a space where the classification is performed by moving axis aligned separation planes. Rezk-Salama et al. [RSKK06] created models of transfer functions that are carefully adjusted by specialists for several data sets of the same type in order to reveal the desired structures. Then, they applied PCA to represent the parameter set of each model by a single variable with an associated semantic. The models can be reused for new data sets by setting only that variable. Ma et al. [MWT*98] used SOMs to extract a reduced number of representative multi-dimensional voxel signatures from volume data. They built probabilistic classification functions based on these signatures.

Regarding the actual representation of TFs for rendering, Kniss et al. [KPI*03] discussed the high memory requirements to store high-dimensional TF lookup tables and proposed Gaussian multi-dimensional transfer functions, which can be analytically evaluated. In another work [KUS*05], Kniss et al. performed volume visualization by separating data classification from the transfer function. Classification is done by using a probabilistic approach, while the TF assigns optical properties to classes considering uncertainty.

One may think of our method as a non-discrete classification scheme (dimensional reduction) combined with transfer function design in an easily manageable space. Since we use self-organizing maps to reduce the dimension of the interaction space, we briefly review this topic in the next subsection.

2.2. Self-Organizing Maps

Kohonen's maps [Koh97] are regular structures containing cells and neighborhood relations between them (in this work, a 2D grid topology is employed). Each cell contains an nD weight vector that represents a subset (class) of the mapped data. The map is built through an iterative unsupervised learning process in which the training cases are the multi-dimensional values to be mapped. For each presented training case, the winner cell (BMU) — the cell with the most similar weight vector — is identified (Equation 1). Then, the current weight vectors of this cell ($\vec{W}_t$) and of the cells in a limited neighborhood are modified. Each new weight ($\vec{W}_{t+1}$) vector is obtained through interpolation between the training case and the respective current weight vector (Equation 2). The coefficient of this interpolation for each cell depends on a predefined constant (η) and the neighborhood function (nf), which is one for the winner cell and decreases for the other cells according to their topological distance to the winner cell.

$$\begin{aligned} BMU &= C_i \mid \forall j \in [1,n], \\ &\quad d(\vec{W}(C_i),\vec{T}) \leq d(\vec{W}(C_j),\vec{T}) \end{aligned} \tag{1}$$

$$\begin{aligned} \vec{W}_{t+1}(C_i) &= (1-a)\times\vec{W}_t(C_i)+a\times\vec{T} \\ a &= \eta \times nf(td(BMU,C_i)) \end{aligned} \tag{2}$$

BMU is the winner cell (or best matching unit), C_i and C_j are map cells, n is the number of cells, d is a distance metric between vectors, $\vec{W}$ is a function that produces the weight vector of a cell and $\vec{T}$ is a training case. $\vec{W}_{t+1}$ is the new weight vector of a cell, $\vec{W}_t$ is the current weight vector, η is a constant between zero and one, nf is the neighborhood function and td is the topological distance between two cells. Usually, the distance metric is the weighted Euclidean distance, i. e., vector elements are multiplied by weights before the evaluation of Euclidean distance. The weights define the importance of each vector element.

The weight vectors are initialized with random values between zero and one. Then, the training cases are presented to the map in random order until it converges, so each training case is often presented to the map more than once. The balance between convergence time and map stability is determined by the value of η. The weight vectors must have the same dimension of the input data.

Spherical maps have the same features, except that they present a spherical topology that is built by subdividing the triangular faces of an icosahedron in order to obtain more nodes. In this work we use self-organizing maps to represent all voxels of a volume data set in a two-dimensional space. We build the SOMs using, as training cases, multi-dimensional voxel signatures obtained from the data set.

3. Design of Multi-dimensional Transfer Functions

Our multi-dimensional transfer function design technique consists basically of three processes: building of a two-dimensional map from the voxels' multi-dimensional signatures; dimensional reduction of the nD signatures, where each signature is replaced by its coordinates in the map space; and specification of a transfer function in the reduced space — the 2D map. Therefore, the actual multi-dimensional transfer function is the composition of two functions: the dimensional reduction function and the transfer function in the map space.

Self-organizing maps have interesting properties for dimensional reduction. They can represent a set of multi-dimensional values (for example, a set of voxel signatures) in a compact way as well as group the values by similarity, according to a distance metric. Consequently, similar signatures have similar map coordinates. This feature facilitates the map exploration and understanding. Additionally, since the screen is a two-dimensional space, the exploration of 2D maps for TF design can be very natural.

3.1. Building of maps

The map building process starts with a preprocessing phase, when complex voxel signatures (such as derivatives and statistical measures) are extracted from the volume data and normalized. This way, each voxel has an nD signature (a set of scalar values represented as a vector) that can be used as a training case for the map building algorithm. It is important to mention that, depending on the source of volume data, there are many background voxels, which do not carry useful information (air around scanned objects in CT/MRI volume data, for example), and would influence the map due to their high occurrence. Upon user decision, they can be partially removed from the input set of the training process by a very simple region growing technique using as seeds the voxels identified as background in the most exterior regions of the volume.

The signatures of all non-background voxels are employed as training cases and presented in random order to the self-organizing map, and two types of neighborhood functions are applied. In a first stage we define the overall aspect of the map by training it using a Gaussian neighborhood function (Equation 3), which depends on the topological distance from the winner cell to the cell in focus. Next, we continue the map training with a modified neighborhood function (Equation 4) that also depends on the distance (d) between the training case and the weight vector of the winner cell (refer to Subsection 2.2). This modified neighborhood function is designed in order to allow a voxel with a signature far from the weight vector of the corresponding winner cell (according to the distance metric) to have more influence on the map. Without this strategy, large homogeneous regions of the volume would tend to dominate the map, while important regions with fewer voxels would be badly represented (signatures far from their respective winner cells).

$$nf1(td) \;=\; \exp\left(-\frac{td^2}{3}\right) \tag{3}$$

$$nf2(td,d) \;=\; \min(d,1)\times\exp\left(-\frac{td^2}{3}\right) \tag{4}$$

We use as topological distance the Euclidean distance between the integer 2D coordinates of two cells in the map grid. For spherical maps, the topological distance is the number of edges in the shortest path connecting two cells. At the end of the process, we have a Kohonen (or spherical) map where each cell has an associated weight vector that represents a class of voxels, being the most similar weight vector for all elements in that class.

3.2. Dimensional Reduction

Dimensional reduction is motivated by the need to provide a simplified space for the design of multi-dimensional transfer functions. When using Kohonen maps, two-dimensional map space coordinates, in the interval from zero to one, can be associated to cells according to their position in the 2D grid. Dimensional reduction can be performed by replacing each voxel signature by the coordinates of its respective winner cell, which has the most similar vectorial signature. However, this would cause unnecessary discretization. To avoid this, we create two multiquadric radial basis functions (multiquadric RBFs) [Buh03], for x and y map space coordinates, based on the weight vectors of all cells. For spherical maps we adopt x, y and z position coordinates ranging from -1 to 1, and use three RBFs to obtain the coordinates of voxel signatures. Thus, the RBFs support the final step in dimensional reduction of voxel signatures by producing, through interpolation, the proper x and y (and z) map coordinates for each nD voxel signature.

Dimensional reduction normally implies loss and distortion of information, but volumetric data usually have properties that reduce this problem. The voxels signatures are usually not uniformly distributed in their domain (they form clusters, which are well represented in the map), and elements of the voxel signatures are often not completely independent [TLMM02]. Moreover, voxel signatures that are not present in the training set do not require space in the map, since SOMs are able to perform non-linear dimensional reduction. This ability and the well-defined topology and shape of SOMs have guided our choice for this particular dimensional reduction scheme.

3.3. Transfer Function Specification

After the dimensional reduction step, the continuous map space defined by the RBFs becomes the TF domain. The user can interactively define the mapping from map coordinates (which represent voxel signatures) to optical properties. We propose an interface for specification of color and opacity transfer functions that provides dual domain interaction [KKH02] as well as visualizations of the transfer function and of the voxel signatures.

3.3.1. Interaction in the TF Domain

The visualization of voxel signatures in our interface is obtained by directly mapping up to three elements of the weight vectors of the map cells to the three color channels. The user decides which elements of the nD signatures must be mapped to each basic color. One element can be associated to more than one color channel and a color channel may have no elements mapped to it. This interface feature illustrates the distribution of voxel signatures on the map and can be used to build color TFs as described below. Figure 1 (a and b) shows the distribution of voxel signatures of the well-known engine data set. The same regions (clusters of signatures) can be found in both maps.

The transfer function (color and opacity) is represented as an RGBA image and displayed by blending it with a checkerboard pattern according to the equation $C_{out} = \alpha_{blending} \times C_{TF} + (1 - \alpha_{blending}) \times C_{checker}$, where colors are represented by C and opacity by α, and $\alpha_{blending} = \log(1 + s \times \alpha_{TF}) / \log(1 + s)$. s is a constant greater than one (we empirically use 200) that allows TFs with small opacities to be clearly visualized. Figure 1 displays TFs on a Kohonen map (c) and on a spherical map (d) generated for visualizing the engine data set. The rendering of the data set using the TF in Figure 1c is shown in Figure 3a.

Transfer functions are composed by blending several 2D Gaussian opacity TFs, each one having an associated 2D color TF. We provide three types of color TFs that can be associated to a Gaussian opacity function: a constant color chosen from a colorpicker; map coordinates directly mapped to color channels; and elements of weight vectors of map cells mapped to color channels (for each map coordinate the weight vectors of the near cells are interpolated and mapped to colors). At each step a new Gaussian TF is specified and then blended with the current TF according to Equations 5 and 6, for opacity and color, respectively. The result becomes the current transfer function and the composition continues until the desired TF is reached. At start, the current TF has zero opacity and RGB colors for all the map space.

$$\alpha_{new} = 1 - (1 - \alpha_{current}) \times (1 - \alpha_{Gaussian}) \qquad (5)$$

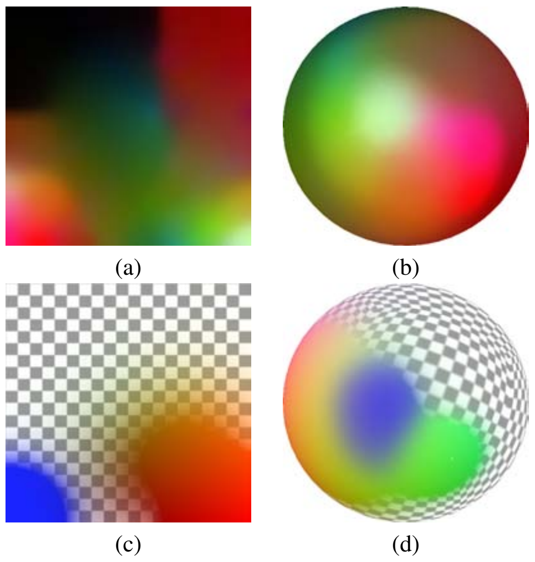

Figure 1: *Maps of three-dimensional voxel signatures — a Kohonen map (a) and a spherical map (b). Scalar value is mapped to red, gradient magnitude to green and directional second derivative to blue. Transfer functions displayed on a Kohonen map (c) and on a spherical map (d).*

$$C_{new} = \frac{\alpha_{current} \times C_{current} + \alpha_{Gaussian} \times C_{Gaussian}}{\alpha_{current} + \alpha_{Gaussian}} \qquad (6)$$

In our interface, by clicking or dragging the mouse on the map representation, the user moves a circle whose center is the peak of a Gaussian function and whose radius is its standard deviation (σ). The Gaussian opacity TF — defined by Equation 7, where d is the distance to the center of the circle — is scaled by a constant k, between zero and one, which is linearly mapped to the circle color, with blue being zero and red being one. The parameters σ and k can be increased or decreased using the keyboard. In order to fully explore the spherical maps, they can be rotated by dragging the mouse using the right button.

$$\alpha_{Gaussian} = k \times \exp\left(-\frac{d^2}{2\sigma^2}\right) \qquad (7)$$

The transfer function used for rendering is the composition of the current color and opacity TF and the Gaussian TF represented by the circle. This scheme provides interactive previewing of the effect of the composition while the user explores the map by moving the circle on it. When the desired effect is reached, the user can set the composition as the current TF using the space bar, and other Gaussian function can be further experimented. Our interface keeps track of all transfer functions defined during a session, and provides a tree representation of this evolution using static thumbnails of the volume rendered with the corresponding TF. This al-

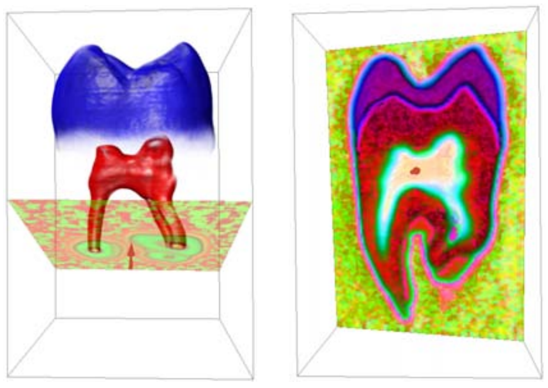
(a) (b)

Figure 2: *Visualizations of the tooth data set: a semi-transparent slice blended with the tooth image rendered using a transfer function specified in a 2D space built from a Kohonen map (a); and a fully opaque slice of the tooth colored according to voxel coordinates in a spherical map (b). The noisy regions can be clearly seen. The red arrow is the plane normal.*

lows simple recovering of previous TFs by clicking on the thumbnails.

3.3.2. Interaction in the Spatial Domain

At any time the user can rotate and translate the volume and place a clipping plane to better explore inner structures. The volume slice defined by the clipping plane is textured by mapping to color channels the map coordinates of the voxels sampled by the slice. This causes an interesting coloring effect that helps in inspecting the volume. The slice is blended with the rendered volume using an opacity value controlled by the user, as shown in Figure 2. When Kohonen maps are employed, the x and y map space coordinates of the voxels are mapped to red and green. When using a spherical map, the x, y and z map space coordinates are mapped to RGB colors. The user can also click on the clipping plane to set the position of the Gaussian function peak to the map coordinates of the voxel pointed by the mouse cursor, emphasizing this region. By moving the mouse on the clipping plane, the user can see the position of the pointed voxel depicted as a white cross in the map graphical representation. This spatial domain interaction mapped to TF domain helps in understanding the relationship between both domains.

4. Implementation Aspects

We implemented map training and dimensional reduction as offline processes, but rendering and transfer function specification demand interactive rates, which are achieved through an intensive use of the GPU.

The map coordinates of the voxels are stored in a 3D RGB texture, which is sampled using view-aligned slices as proxy geometry. When using a Kohonen map, the transfer function is stored in a 2D RGBA texture which is accessed by using the R and G components (the x and y map coordinates) of the sampled 3D RGB texture. The blue component is used to identify background (zero) or non-background (one) voxels. Background voxels must receive zero opacity during rendering since they are not well represented in the map. Nevertheless, due to hardware interpolation, the blue component can assume values between zero and one. With this in mind, the opacity is actually modulated by a smoothed step function of the blue component. When using a spherical map, the TF is stored in the GPU memory as an RGBA cube map and is accessed using the RGB values of the 3D texture, taken as vectors (the value of each color channel is first converted to the interval $[-1, 1]$. Background voxels have null vectors and the opacity is modulated by a smoothed step function of the $L2$-norm of the vectors. The blending of TFs and the evaluation of Gaussian opacity functions also run in GPU.

When sampling the three-dimensional texture for rendering, interpolation must be performed. The hardware can automatically interpolate the map coordinates stored in the 3D texture and generally this produces good results. However, in our approach, it is more correct to interpolate color and opacity associated to voxels (see [HBH03] for better understanding). In our implementation, we use the GPU to create another 3D texture, with the same size, containing the RGBA values that result from the evaluation of the transfer function for each voxel, and this texture is sampled for rendering. When the transfer function changes, this texture must be recomputed, but this strategy is fast enough for our purposes. We also calculate another 3D RGB texture to store the gradient field of the opacity. This is done in GPU by applying central differences on each voxel. The opposite vector of the gradient of the opacity is used as surface normal for shading. Since we are using complex signatures for each voxel, this scheme for evaluating normals is more accurate than sampling a 3D texture containing the precomputed normals of the scalar field. Additionally, the normals of the opacity field do not have ambiguity in their orientation (see [LM04]). In our implementation, we set hardware interpolation of map coordinates and precomputed normals as default options, but the user can select color and opacity interpolation and normals computed on the fly as high-quality rendering options.

As for the RBF design, we solve the systems of equations with the Lapack library [ABB*99]. GLUT and the GLUI libraries are used for the interface, while the rendering is based on OpenGL and Cg, with the framebuffer objects extension of OpenGL used in hardware-accelerated computing.

5. Results and Discussion

We tested our method using well-known data sets (see Figure 3), comprising scalar and multivariate volume data.

Similar results were obtained using Kohonen and spherical maps. Most of the data sets were successfully visualized using voxels signatures based on scalar value, gradient magnitude and directional second derivative. For noisy scalar data, however, we achieved better results using statistical signatures, such as mean scalar value, standard deviation and cubic root of the third-order statistical moment, taken from a small subvolume centered at the voxel under focus. Figure 3 shows visualizations of test data sets obtained using different sets of voxel attributes as signatures. In all these renderings, we used the automatic generated color transfer functions (see Subsection 3.3.1). For the hurricane data set we used only the colormap which assigns voxel attributes to color channels, since the attributes carry a clear physical meaning: temperature was mapped to red, pressure to green and wind speed to blue. However, the tooth data set (Figure 2) was visualized using manually chosen colormaps and statistical signatures, achieving a very good separation of the pulp. Since self-organizing maps group similar voxel signatures, the automatic generated color transfer functions produce very good results because they assign different colors to different regions of the map, which correspond to voxels with considerably different attribute's values. Regarding our shading using normals computed on the fly, results can be seen in Figure 3 (b, d and f). It is worth to mention that for the multivariate data sets, such as the hurricane, we can not use meaningful precomputed normals.

The importance of each voxel attribute is defined by weights (see Subsection 2.2). We suggest associating smaller weights with higher-order voxel attributes. The visualizations presented in this paper were produced using weights of 1.3, 1.0 and 0.7 for the statistical variables formerly mentioned, respectively, and the same weights for scalar and first and second derivative values, respectively. For the hurricane data set the weights were 1.0 for wind speed and pressure, and 0.5 for temperature.

Due to the loss and distortion of information usually caused by dimensional reduction, our method can not provide accurate quantitative information about the volume data during the transfer function specification. However, our approach is well suitable for revealing qualitative aspects such as shape of structures and dissimilarity between regions. By moving the Gaussian opacity function on the map space (see Subsection 3.3.1), the user quickly obtains an overview of the main structures in the data volume. After, by carefully tuning the parameters of the Gaussian functions and combining them, meaningful visualizations can be built. Automatically generated color transfer functions are usually a good choice, which turns the design process less difficult (for example, the blood vessels in the Sheep Heart data set can be easily emphasized using our colormaps). The history tree, briefly described in Subsection 3.3, provides a powerful mechanism for exploring the transfer function domain, allowing not only "undo" and "redo" operations, but navigation in the whole history of TF modifications.

Map type	Training steps/Time	
	Neigh. function 1	Neigh. function 2
2D map	3,000,000/57	4,500,000/87
Sph. map	3,000,000/151	5,250,000/264

Table 1: *Number of training steps and training time (seconds) for both types of maps in the two stages of training, which use different neighborhood functions.*

Regarding the building of the self-organizing maps for dimensional reduction, it involves setting of parameters and selection of voxel attributes to be mapped. In the following we provide guidelines for choosing appropriate parameters values and discuss their impact on processing time.

Self-organizing maps easily converge, becoming stable along the training, for small values of η (a constant between zero and one), and large number of training steps, but one generally has to assume a compromise between stability and training time. Fortunately, in our method, map stability is not a critical point. The lower bound of the number of training steps needed to guarantee stability depends on the map size, the neighborhood function, the value of η used in training (see Subsection 2.2), and the characteristics of the mapped data. Experimentally we found that good results can be achieved using Kohonen maps with more than 20×20 cells or spherical maps with more than seven subdivisions by icosahedron edge. For building the map, first we set η to 0.3 and perform training using the neighborhood function described in Equation 3 (see Subsection 3.1). Then we refine the map by training it using 0.5 as η value and Equation 4 as neighborhood function. The values for η were also chosen based on experiments.

The time needed to train the maps is quadratically related to the map size (number of cells) and is proportional to the size of the weight vectors. At each training step, the search for the winner cell has linear complexity, being dependent on the number of cells; the number of training steps needed is also directly related to the map size. The size of the weight vectors have linear influence on the time spent to search for the winner cell and update weight vectors (see Subsection 2.2). The time spent in the dimensional reduction step (see Subsection 3.2) is proportional to the number of non-background voxels (see Subsection 3.1), the size of the map, and the size of weight vectors. The RBF evaluation is performed for each non-background voxel and its time consumption is proportional to the number of cells and the size of weight vectors.

The results presented herein were obtained using Kohonen maps with 32×32 cells and spherical maps with ten subdivisions by icosahedron edge. Table 1 shows the number of performed training steps and the training time for both map types. Spherical maps usually require little more training; and the training steps are slower, mainly because

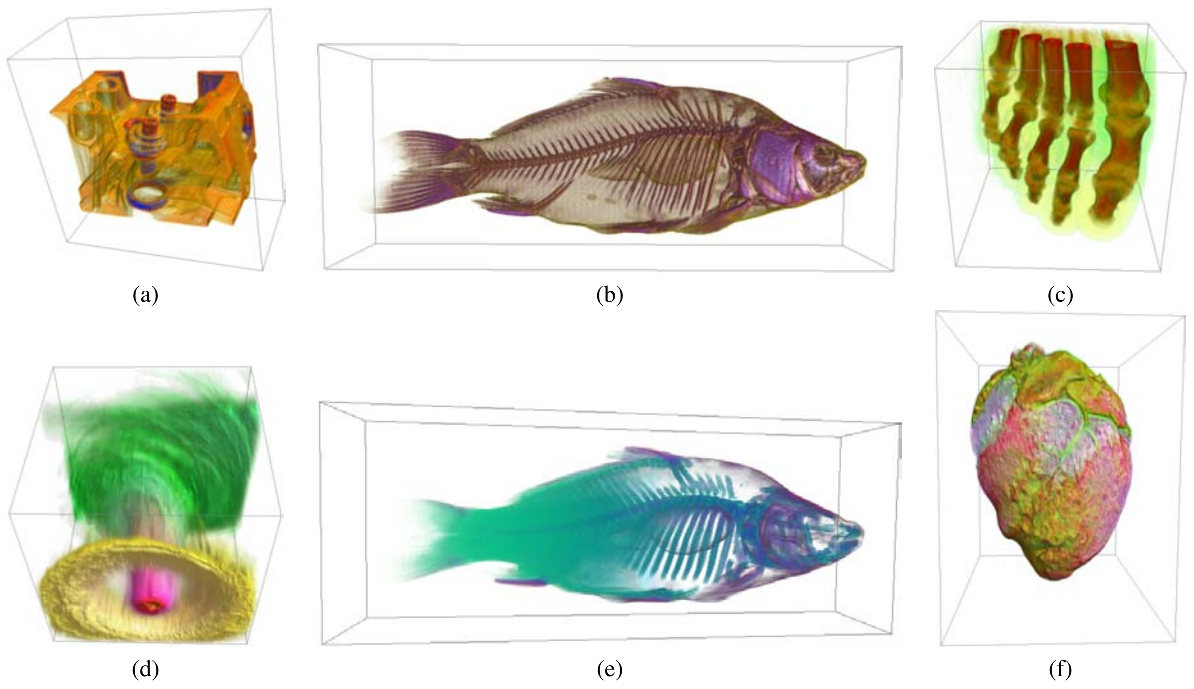

Figure 3: *Visualizations obtained using Kohonen maps — (a) engine data set and (b) carp data set, both using scalar and derivative values (gradient magnitude and directional second derivative) as voxel signature; (c) foot data set, using statistical signatures (mean scalar value, standard deviation, cubic root of the third-order statistical moment); and (e) carp data set, using the normalized z coordinate of the voxels and same three statistical signatures, thus composing a four-dimensional voxel signature (z axis is horizontally represented) — and Spherical maps — (d) hurricane data set at the 24th time step, using wind speed, pressure and temperature as voxel signature; and (f) sheep heart data set, using the same statistical signatures as (c).*

Data set	Size	Non-B. voxels	Time
Engine	$256^2 \times 128$	5297265	140
Tooth	$128^2 \times 256$	4193888	110
Foot	256^3	4890753	133
Carp	$256^2 \times 512$	32806605	845
Hurricane	$256^2 \times 128$	8388608	220
Sheep heart	256^3	16776721	375

Table 2: *Size, number of non-background voxels and time (seconds) spent in the dimensional reduction step, using Kohonen maps and three voxel attributes, for the data sets presented in this paper.*

the search for neighbors of the winner cell, for weight update, is more complex. Table 2 shows size, number of non-background voxels and dimensional reduction time, using Kohonen maps, of the visualized data sets. Using spherical maps the times were about thirteen percent greater. The time needed to extract high-order signatures from scalar volumes is short and is not considered because this is done only once for each data set. Our results were generated using an AMD Athlon 64 3700+, 2.21GHz CPU and 2GB of RAM.

6. Conclusions and Future Work

In this paper we presented a new approach for the design of multi-dimensional transfer function that uses self-organizing maps to perform dimensional reduction of the voxel attributes. The strongest points of our technique are simplicity and flexibility. Our approach allows building multi-dimensional transfer functions through the exploration of a simplified (reduced) space where traditional interaction techniques can be employed. A simple and effective interface for transfer function design is provided, and the user can interact with the system in both spatial and TF domains.

Self-organizing maps have the ability of representing clusters of voxel signatures in a compact way, and this helps to understand the data distribution. All relevant voxel signatures are represented in the map and every region of the map has voxels mapped to it. Moreover, to explore two-dimensional maps is easier and faster than to navigate in a class hierarchy. The proposed dimensional reduction scheme requires a preprocessing step, but it has clear advantages in relation to volume segmentation techniques because it performs a non-discrete classification which can represent uncertainty. In addition, with simple interaction, the user can change the transfer function defined in the map space, interactively obtaining new visualizations. Our automatically generated color TFs are able to emphasize details that are hard to be visualized using other approaches. When compared to other visualization techniques based on transfer functions, ours has the advantage of having a powerful built-in classification scheme. The existence of classes and class properties is, however, kept abstract. As a drawback, our approach suffers from loss of information, caused by the dimensional reduction process.

As future work, we intend to assess the usability of our technique and to experiment our method on visualizing time-varying multivariate volume data. Coherence of time-varying transfer functions could be maintained by incremental training of the SOMs for each time step; and the evaluation time of the RBFs can be reduced using approximating RBFs. We also want to transport transfer functions designed in map space to the actual multi-dimensional space using the Gaussian multi-dimensional TFs proposed by Kniss [KPI*03]. Another promising future work is the semi-automatic search for important structures in the map. This search could be aided by an interface that would provide additional information about the spatial distribution of voxels. At last, it is important to emphasize that we have built a map of voxel signatures for each visualized data set. The success of performing dimensional reduction of a data set using a map built from another but similar data set needs to be assessed, and this is also intended as future work.

Acknowledgments

This work was supported by CNPq (Brazilian Council for Research and Development). We thank Fausto R. Blanco, for helping with the RBFs, and the UFRGS' CG group.

References

[ABB*99] Anderson E., Bai Z., Bischof C., Blackford S., Demmel J., Dongarra J., Du Croz J., Greenbaum A., Hammarling S., McKenney A., Sorensen D.: *LAPACK Users' Guide*, third ed. 1999.

[Buh03] Buhmann M. D.: *Radial Basis Functions: Theory and Implementations*. 2003.

[HBH03] Hadwiger M., Berger C., Hauser H.: High-quality two-level volume rendering of segmented data sets on consumer graphics hardware. In *Proceedings of IEEE Visualization* (2003), pp. 40–47.

[HM03] Huang R., Ma K.-L.: Rgvis: Region growing based techniques for volume visualization. In *Proceedings of the 11th Pacific Conference on Computer Graphics and Applications* (2003), pp. 355–363.

[KKH02] Kniss J., Kindlmann G., Hansen C.: Multidimensional transfer functions for interactive volume rendering. *IEEE Transactions on Visualization and Computer Graphics 8*, 3 (2002), 270–285.

[Koh97] Kohonen T.: *Self-organizing maps*. 1997.

[KPI*03] Kniss J., Premoze S., Ikits M., Lefohn A., Hansen C., Praun E.: Gaussian transfer functions for multi-field volume visualization. In *Proceedings IEEE Visualization* (2003), pp. 497–504.

[KUS*05] Kniss J. M., Uitert R. V., Stephens A., Li G.-S., Tasdizen T., Hansen C.: Statistically quantitative volume visualization. In *Proceedings of IEEE Visualization* (2005), pp. 37–44.

[KWTM03] Kindlmann G., Whitaker R., Tasdizen T., Möller T.: Curvature-based transfer functions for direct volume rendering: Methods and applications. In *Proceedings of IEEE Visualization* (October 2003), pp. 513–520.

[LM04] Lum E. B., Ma K.-L.: Lighting transfer functions using gradient aligned sampling. In *Proceedings of IEEE Visualization* (2004), pp. 289–296.

[MWT*98] Ma F., Wang W., Tsang W. W., Tang Z., Xia S., Tong X.: Probabilistic segmentation of volume data for visualization using som-pnn classifier. In *Proceedings of IEEE symposium on Volume visualization* (1998), pp. 71–78.

[PLB*01] Pfister H., Lorensen B., Bajaj C., Kindlmann G., Schroeder W., Avila L. S., Martin K., Machiraju R., Lee J.: The transfer function bake-off. *IEEE Computer Graphics and Applications 21*, 3 (2001), 16–22.

[RSKK06] Rezk-Salama C., Keller M., Kohlmann P.: High-level user interfaces for transfer function design with semantics. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 1021–1028.

[SK02] Sangole A., Knopf G. K.: Representing high-dimensional data sets as closed surfaces. *Information Visualization 1*, 2 (2002), 111–119.

[TLM01] Tenginakai S., Lee J., Machiraju R.: Salient iso-surface detection with model-independent statistical signatures. In *Proceedings of IEEE Visualization* (2001), pp. 231–238.

[TLM05] Tzeng F.-Y., Lum E. B., Ma K.-L.: An intelligent system approach to higher-dimensional classification of volume data. *IEEE Transactions on Visualization and Computer Graphics 11*, 3 (2005), 273–284.

[TLMM02] Takanashi I., Lum E. B., Ma K.-L., Muraki S.: Ispace: Interactive volume data classification techniques using independent component analysis. In *Proceedings of the 10th Pacific Conference on Computer Graphics and Applications* (2002), pp. 366–374.

[TM04] Tzeng F.-Y., Ma K.-L.: A cluster-space visual interface for arbitrary dimensional classification of volume data. In *Proceedings of the symposium on Data visualisation* (2004), pp. 17–24.

[vBG06] Šereda P., Bartroli A. V., Gerritsen F. A.: Automating transfer function design for volume rendering using hierarchical clustering of material boundaries. In *Proceedings of IEEE/EuroGraphics Symposium on Visualization* (2006), pp. 243–250.

[vBSG06] Šereda P., Bartroli A. V., Serlie I. W. O., Gerritsen F. A.: Visualization of boundaries in volumetric data sets using lh histograms. *IEEE Transactions on Visualization and Computer Graphics 12*, 2 (2006), 208–218.