Efficient Volume Exploration Using the Gaussian Mixture Model

Yunhai Wang, Wei Chen, *Member*, *IEEE*, Jian Zhang, Tingxing Dong, Guihua Shan, and Xuebin Chi

Abstract—The multidimensional transfer function is a flexible and effective tool for exploring volume data. However, designing an appropriate transfer function is a trial-and-error process and remains a challenge. In this paper, we propose a novel volume exploration scheme that explores volumetric structures in the feature space by modeling the space using the Gaussian mixture model (GMM). Our new approach has three distinctive advantages. First, an initial feature separation can be automatically achieved through GMM estimation. Second, the calculated Gaussians can be directly mapped to a set of elliptical transfer functions (ETFs), facilitating a fast pre-integrated volume rendering process. Third, an inexperienced user can flexibly manipulate the ETFs with the assistance of a suite of simple widgets, and discover potential features with several interactions. We further extend the GMM-based exploration scheme to time-varying data sets using an incremental GMM estimation algorithm. The algorithm estimates the GMM for one time step by using itself and the GMM generated from its previous steps. Sequentially applying the incremental algorithm to all time steps in a selected time interval yields a preliminary classification for each time step. In addition, the computed ETFs can be freely adjusted. The adjustments are then automatically propagated to other time steps. In this way, coherent user-guided exploration of a given time interval is achieved. Our GPU implementation demonstrates interactive performance and good scalability. The effectiveness of our approach is verified on several data sets.

Index Terms—Volume classification, volume rendering, Gaussian mixture model, time-varying data, temporal coherence.

1 Introduction

Volume exploration focuses on revealing hidden structures in volumetric data sets. Effective exploration is a challenging problem because there is no prior information available with respect to data distribution. This difficulty is magnified by the fact that exploring and manipulating in three-dimensional (3D) space is typically counterintuitive and laborious. Feature spaces (the axes of which represent attributes of the data) are usually used to design transfer functions. With a properly designed feature space, transfer function design becomes a user controllable process that structures the feature space and maps selected data properties to specific colors and opacities. To understand these various structures better, a number of multidimensional transfer function design schemes have been proposed. In particular, two-dimensional (2D) transfer functions [15] based on scalar values and gradient magnitudes are very effective in extracting multiple materials and their boundaries. The specification of 2D transfer functions can be performed with the help of various classification widgets. However, the selection of features within the 2D feature space is a trial-and-error process and is very likely to yield unsatisfactory results. The gap between the flexibility of multidimensional transfer function design and the fidelity requirement of volume exploration makes transfer function design challenging. For time-varying data, additional care should be taken to preserve coherence among different time steps as well as reduce the computational cost of per-step exploration.

We have identified three reasons for the difficulty in the multidimensional transfer function design. First, the search space is very large. The user is often required to spend much time understanding the underlying features and their spatial relationships. Second, modulating the parameters of classification widgets to maximize the likelihood of feature separation is not trivial, even when all features have been identified. Third, traditional classification widgets (e.g., rectangular and triangular) are too regular to describe multidimensional features, which may have complex shapes.

In a previous paper [37], we introduced a novel volume exploration scheme by approximating the exploration space with a set of Gaussian functions. This scheme takes an analyze-and-manipulate approach. Prior to manipulation, it performs a maximum likelihood feature separation of the feature space to construct a continuous and probabilistic representation using the Gaussian mixture model (GMM). The GMM enables semiautomatic volume classification by converting mixture components to a set of suggestive elliptical transfer functions (ETFs). Here, "semiautomatic" means that the number of mixture components is determined by the user, and that the suggested ETFs may be adjusted.

- *Y. Wang, J. Zhang, G. Shan, and X. Chi are with the Supercomputing Center, Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China. E-mail: cloudseawang@gmail.com, {zhangjian, sgh, chi}@sccas.cn.*
- *W. Chen is with the State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310058, China. E-mail: chenwei@cad.zju.edu.cn.*
- *T. Dong is with the Department of Electrical Engineering and Computer Science, The University of Tennessee, Knoxville, TN 37996. E-mail: tingxingdong@gmail.com.*

Manuscript received 1 June 2010; revised 10 Apr. 2011; accepted 26 Apr. 2011; published online 10 June 2011.
Recommended for acceptance by H.-W. Shen, J.J. van Wijk, and S. North.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCGSI-2010-06-0112.
Digital Object Identifier no. 10.1109/TVCG.2011.97.

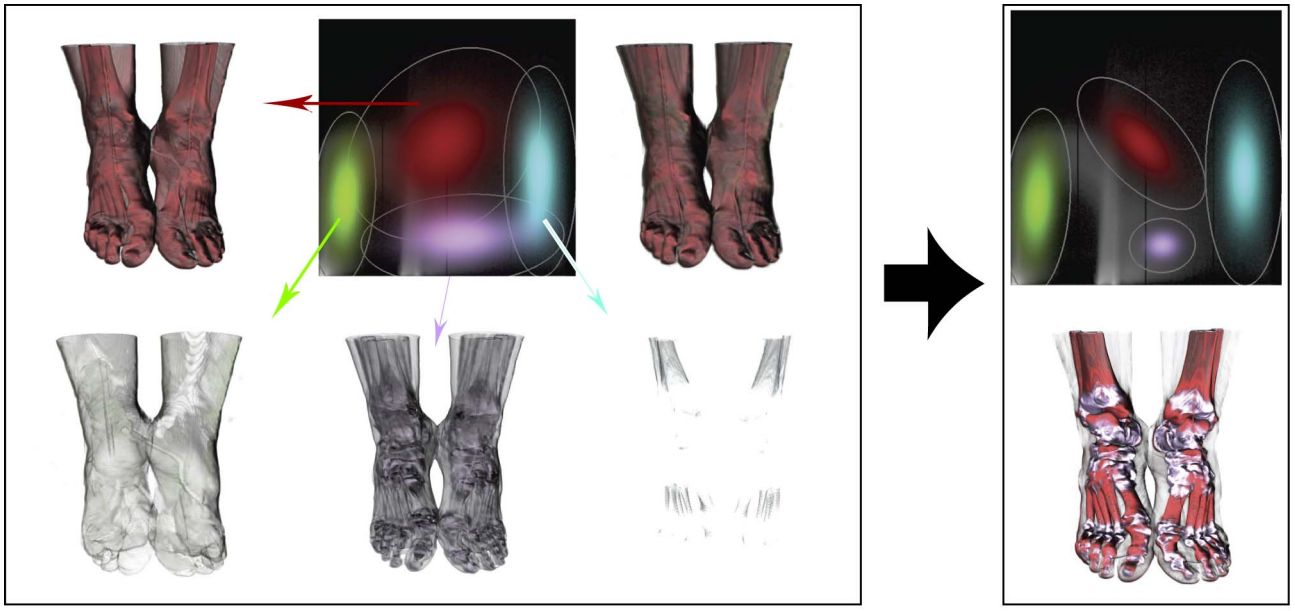


Fig. 1. Modeling the density and density gradient magnitude feature space using the GMM for the Feet data set. (a) Automatically generated ETFs, the volume rendering, and individual volume renderings associated with each ETF. (b) Result obtained by scaling the ETFs in red and plum, and adjusting their maximum opacities with the phalanges and ankles clearly shown.

The obtained ETFs not only facilitate preintegrated volume rendering but also give rise to a suite of flexible elliptical classification widgets. Although achieving a satisfactory classification still requires user adjustments to the ETFs, the interaction is effective, as demonstrated in Fig. 1.

This paper enhances the GMM-based volume classification scheme by employing an incremental GMM estimation algorithm [3] to ease classification of time-varying data. The incremental algorithm estimates the GMM of each time step by exploiting the GMM generated from previous steps. Compared with previous work [19], [38] that deals with the collection of all time steps, incrementally estimating the GMM processes the sequence of time steps individually in less memory and time.

As the data set itself may be noisy, the results provided by incremental GMM estimation can typically be improved by appropriate user adjustments. Making these adjustments is actually a procedure of choosing the features of interest and feeding this prior knowledge into the visualization. However, adjusting the time steps individually imposes a large workload upon the user, and may result in a temporally incoherent classification. Conversely, directly applying user adjustments to the initial ETFs of other time steps may not catch the variations in the feature space and may produce results with temporal incoherence. Assuming that the density distribution variations among the time steps reflects the evolution of features, we propose a coherent adjustment propagation technique to solve this problem.

The rest of this paper is organized as follows: The related work is summarized in Section 2. The classification and exploration of a data set using the GMM is described in Section 3. Section 4 introduces the coherent classification of time-varying data. The implementation details are described in Section 5. The results are demonstrated in Section 6. Finally, conclusions are presented in Section 7.

2 Related Work

Related work falls into three categories: 1) transfer function design, 2) time-varying data classification, and 3) Gaussian mixture models.

2.1 Transfer Function Design

A complete review of transfer function design is beyond the scope of this paper; we refer the reader to Pfister et al. [21]. We restrict our discussion to the design of multidimensional transfer functions [18]. Despite their excellent performance in material classification, multidimensional transfer functions did not receive widespread attention until the ground breaking work by Kindlmann and Durkin [13]. Their research shows that determining multidimensional transfer functions in a 2D histogram of data values and gradient magnitudes can effectively capture boundaries between materials. To facilitate the specification of multidimensional transfer function, Kniss et al. [15] introduce a set of manipulable widgets. Local [14], [25] and global information [6], [7] can be incorporated into multidimensional feature spaces as well.

Based on an analysis of the data set itself, many methods have been proposed to simplify the creation of multidimensional transfer functions. Fujishiro et al. [9] and Zhou and Takatsuka [41] utilize topology analysis to automate transfer function generation. Tzeng and Ma [32] use the ISODATA algorithm to perform clustering in multidimensional histograms. Roettger et al. [23] propose transfer functions that consider spatial information in the process of clustering 2D histograms. To structure the feature space effectively, Selver and Güzelis [26] use a self-generating hierarchial radial basis function network to analyze volume histogram stacks. Likewise, Maciejewski et al. [19] apply a nonparametric density estimation technique. Using machine learning techniques, Tzeng et al. [31] introduce a

painting interface to derive high-dimensional transfer functions. Salama et al. [24] derive semantic transfer functions by utilizing principle component analysis. In this paper, we introduce a new volume exploration scheme, by analyzing the feature space using the GMM to maximize the likelihood of feature separation. Immediate visual feedback is enabled by mapping these Gaussians to ETFs and analytically integrating the ETFs in the context of a preintegrated volume rendering process.

2.2 Time-Varying Data Classification

The main challenge in designing transfer functions for time-varying data is that the user has to consider not only the evolution of features, but also temporal coherence. Several effective methods have been developed to address this challenge. Jankun-Kelly and Ma [12] propose to generate summarized transfer functions by merging the transfer functions of all time steps. Tzeng and Ma [33] introduce a solution to compute transfer functions automatically for all time steps, given several transfer functions defined for key time steps. By brushing a 2D time histogram, Akiba et al. [1] classify multiple time steps simultaneously.

Nonetheless, even using time histograms, finding an appropriate transfer function for each time step is still very time consuming. To alleviate this problem, some research has been devoted to semiautomatic classification. Woodring and Shen [38] first utilize temporal clustering and sequencing to find dynamic features and create the corresponding transfer functions. By treating time-varying 2D histograms as a 3D volume, Maciejewski et al. [19] cluster the volume using kernel density estimation to generate transfer functions for all steps. For long time series, these clustering methods will take much time [38]. To resolve this issue, we adopt an incremental clustering method [3] that locally clusters the data of each time step with the clustered result generated from previous time steps. Without collecting all the time steps, our method requires little memory and time. This is similar to the feature tracking approach [27], which uses the detected features of the current step to predict the features of the next step. To preserve temporal coherence, Tikhonova et al. [30] apply a global transfer function to the intermediate representation of the rendered image from each time step. We employ an alternative solution capable of directly generating coherent visualization while generating a transfer function for each time step.

2.3 Gaussian Mixture Models

The GMM [2] is well suited to modeling clusters of points. Each cluster is assigned a Gaussian, with its mean somewhere in the middle of the cluster, and a standard deviation that measures the spread of that cluster. The GMM has been widely used in pattern recognition [29] and medical image segmentation [39]. Recently, the GMM has been introduced to the visualization community by Correa et al. [5] to model uncertainty distributions. In time-critical applications such as neural signal monitoring, data sets are generated on the fly. Hence, modeling an entire data set using the GMM is usually impractical for large time-varying data sets. Accordingly, we employ an incremental GMM estimation algorithm [3], which models the current time step by using estimated GMM parameters generated from previous steps.

3 Exploring Feature Space Using the GMM

The GMM is an unsupervised clustering method. It can extract coherent regions in feature space and corresponding meaningful structures in the input data space [2], where each region is represented by a Gaussian distribution.

We choose the GMM to explore the 2D feature space of volume data for three reasons. First, clustering is achieved by maximizing the likelihood of feature separation. This provides the user with a solid starting point for volume exploration. Second, mixture components can be mapped to ETFs, facilitating a fast preintegrated volume rendering process. Third, the ETFs can be controlled by flexible elliptical classification widgets. GMM-based volume exploration takes an analyze-and-manipulate approach, as shown in Fig. 1. In the analysis stage, the user is provided with a reasonable base for volume classification. In the manipulation stage, the user adjusts the features with the help of flexible classification widgets.

3.1 Maximized Likelihood Feature Separation

Given a 2D feature space (e.g., scalar values and gradient magnitudes), each Gaussian function represents a homogenous region whose corresponding probability distribution function is defined as

$$g(x|\mu,\Sigma) = \frac{1}{2\pi|\Sigma|^{1/2}} e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}, \quad (1)$$

where x is a vector in the feature space, μ is the center vector of the Gaussian, Σ is the 2×2 covariance matrix used to scale and rotate the Gaussian, and $g(x)$ is the probability of x. As such, the distribution of all regions is represented by the Gaussian mixture distribution

$$p(x|\theta) = \sum_{j=1}^{k} \alpha^j g^j(x|\mu^j,\Sigma^j), \quad (2)$$

where α^j is the prior probability of the jth Gaussian and satisfies the following condition:

$$\sum_{j=1}^{k} \alpha^j = 1 \text{ and } \alpha^j \geq 0 \text{ for } j \in \{1,\ldots,k\}. \quad (3)$$

θ denotes the parameter set of the GMM with k components $\{\alpha^j,\mu^j,\Sigma^j\}_{j=1}^k$. As each region corresponds to a feature in the underlying data set, determining the appropriate parameters θ can be converted to a problem of specifying the feature to which the points in the feature space most likely belong. Assuming that the vectors in the feature space $\{x^1,\ldots,x^n\}$ are independent identically distributed, the maximum likelihood estimation of θ is

$$\hat{\theta} = \arg\max_{\theta} p(x^1,\ldots,x^n|\theta) = \arg\max_{\theta} \prod_{i=1}^{n} p(x^i|\theta), \quad (4)$$

where n is the number of voxels in the volume.

As a general method for finding the maximal-likelihood estimation of the parameters of the underlying distribution, the well-known EM algorithm [2] provides an iterative means to determine θ. Given an initial estimated parameter set θ, the EM algorithm iterates over the following steps until it converges to a local maximum of the likelihood function:

- E Step:

$$P(j|x^i) = \frac{\hat{\alpha}^j g(x^i|\hat{\mu}^j, \hat{\Sigma}^j)}{\sum_{j=1}^{k} \hat{\alpha}^j g(x^i|\hat{\mu}^j, \hat{\Sigma}^j)}. \tag{5}$$

- M Step:

$$\begin{aligned} \hat{\alpha}^j &= \frac{1}{n}\sum_{i=1}^{n} P(j|x^i) \hat{\mu}^j = \frac{\sum_{i=1}^{n} P(j|x^i) x^i}{\sum_{i=1}^{n} P(j|x^i)}, \\ \hat{\Sigma}^j &= \frac{\sum_{i=1}^{n} P(j|x^i)(x^i - \hat{\mu}^j)(x^i - \hat{\mu}^j)^T}{\sum_{i=1}^{n} P(j|x^i)}, \end{aligned} \tag{6}$$

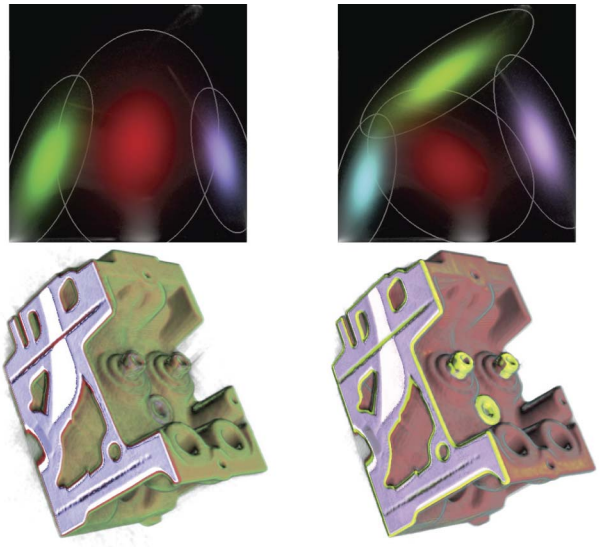
where $i \in [1, \ldots, n]$, $j \in [1, \ldots, k]$, n is the number of voxels in the volumetric data set, $P(j|x^i)$ is the probability of the vector x_i belonging to the jth feature, and E_j is the cumulated posterior probability of the jth Gaussian. The E step finds the expected value of the log-likelihood $\sum_{i=1}^{n} \log p(x^i|\theta)$, and the M step finds new parameters that maximize the expectation computed in the E step. The log-likelihood holds due to the monotonicity of the log function. It is used here to deal with the multiplication of a large number of floating point probability values that are in $(0, 1)$. In a few iterations, a locally optimal solution is achieved. However, convergence to a globally optimal solution is not guaranteed, and the number of iterations depends on the initial assigned parameters. At times, the user has to spend much time finding the proper initial parameters and the optimal number of mixture components. This poses difficulties for interactive volume classification.

To reduce the user's workload, we use the greedy EM algorithm [35], which builds models in an adaptive manner. Starting with a single component whose parameters are easily computed, two steps are alternatively performed: adding a new component to the mixture, and updating the complete mixture using the E and M steps until a convergence criterion is met. Using this greedy algorithm, the initial parameters θ do not need to be chosen by the user, making the number of mixture components manageable. If the results are not acceptable, the user can insert new components to update the GMM. Fig. 2 shows the volume classification of the Engine data set (available at URL http://www.volvis.org/) using this greedy algorithm. In Fig. 2b, a new component is added to the clustering shown in Fig. 2a, providing separation of the main body and the wheels. Although there is no known constructive method to find the global maximum, the greedy EM algorithm we adopted locates the global maximum using a search heuristic [35].

After finding an appropriate θ, each pixel in the feature space is associated with a probability vector $p = (p_1, \ldots, p_k)$, where $p_j = g(x|\theta_j)$. With these vectors, the discrete feature space becomes continuous. In contrast with the kernel density estimation [19], the GMM is a semiparametric density estimation technique, where an analytical Gaussian function represents each cluster. This property greatly favors interaction with a set of transformations.

3.2 Elliptical Transfer Functions

One important advantage of GMM-based separation over previous work [23], [32] is that the obtained mixture components can be converted to ETFs



(a) (b)

Fig. 2. Using the greedy EM algorithm to classify the Engine data set with different numbers of mixture components: (a) three; (b) four. In (b), the main body and wheel parts are separated.

$$\rho(x) = \alpha_{\max} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}, \tag{7}$$

where $\alpha_{\max}$ is the maximum opacity and

$$\Sigma^{-1} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \tag{8}$$

where the initial values of b and c are equal. To guarantee that it can be manipulated as an elliptical primitive, Σ^{-1} must satisfy the following condition [40]:

$$(b + c)^2 - 4ad < 0. \tag{9}$$

Compared with the axis-aligned GTF used in Kniss et al. [16] and Song et al. [28] where Σ^{-1} is a diagonal matrix, an ETF is more general and affords more flexible feature separation. We can use the mixing probability of each Gaussian to set the initial $\alpha_{\max}$ for each ETF, because it represents its maximal contribution to the density distribution.

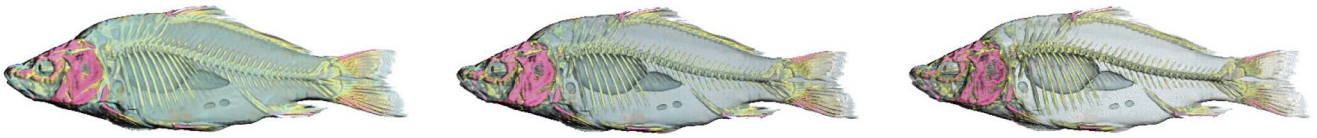
3.3 Preintegrated Volume Rendering with ETFs

Kniss et al. [16] and Song et al. [28] derive analytic forms for preintegrated axis-aligned Gaussian transfer functions. In this section, we demonstrate that an arbitrarily directional ETF can also be incorporated with preintegrated volume rendering.

According to the volume rendering equation [20], opacity can be expressed as

$$\begin{aligned} \alpha &= 1 - e^{-\int_0^D \alpha(\mathbf{x}(\lambda))d\lambda} \\ &= 1 - e^{-\int_0^D \sum_{j=1}^{n} \rho_j(\mathbf{x}(\lambda))d\lambda} \\ &= 1 - e^{\sum_{j=1}^{n} -\int_0^D \rho_j(\mathbf{x}(\lambda))d\lambda}, \end{aligned} \tag{10}$$

where D is the distance between the entry and exit points f and b. By assuming that the feature vector x between $\mathbf{x}_f$ and $\mathbf{x}_b$ varies linearly, the term ρ_j in (10) becomes



(a) (b) (c)

Fig. 3. Visualizing the Carp data set ($256 \times 256 \times 512$) by performing (a) analytic integration with the object sample distance 0.6; (b) numerical integration with the object sample distance 0.3; and (c) numerical integration with the object sample distance 0.6. Their performances are 38, 29, and 54 fps, respectively.

$$\rho_j = \alpha^j_{\max} e^{-\frac{1}{2}\left(\mathbf{x}_f - \mu^j + \lambda\frac{\mathbf{x}_b - \mathbf{x}_f}{D}\right)^T (\Sigma^j)^{-1}\left(\mathbf{x}_f - \mu^j + \lambda\frac{\mathbf{x}_b - \mathbf{x}_f}{D}\right)}.$$

Suppose the feature vector $\mathbf{x}$ consists of a density component and a gradient magnitude component, we have $\mathbf{x} = \{s, g\}$, $\mathbf{x}_f = (s_f, g_f)$, and $\mathbf{x}_b = (s_b, g_b)$. We define $k_s = \frac{s_b - s_f}{D}$, $k_g = \frac{g_b - g_f}{D}$, $d_s = s_f - s_j$, $d_g = g_f - g_j$, yielding

$$\begin{aligned} I_j = {} & \left(ak_s^2 + (b+c)k_s k_g + dk_g^2\right)\lambda^2 \\ & + 2(ak_s d_s + 0.5(b+c)(k_s d_g + k_g d_s) + dk_g d_g)\lambda \\ & + (ad_s^2 + (b+c)d_s d_g + dd_g^2), \end{aligned}$$

where I_j is the exponent of ρ_j. Let

$$\begin{aligned} A &= \sqrt{ak_s^2 + (b+c)k_s k_g + dk_g^2}, \\ B &= (ak_s d_s + 0.5(b+c)(k_s d_g + k_g d_s) + dk_g d_g)/A, \\ C &= \left(ad_s^2 + (b+c)d_s d_g + dd_g^2\right) - B^2, \end{aligned}$$

the integral $-\int_0^D \rho_j(\mathbf{x}(\lambda))d\lambda$ in (10) can be written as

$$\begin{aligned} R_j &= -\int_0^D \alpha^j_{\max}\rho_j(\mathbf{x}(\lambda))d\lambda \\ &= -\int_0^D \alpha^j_{\max} e^{-\frac{1}{2}((A\lambda+B)^2+C)}d\lambda \\ &= -\alpha^j_{\max}\sqrt{\frac{\pi}{2}}\frac{e^{-\frac{C}{2}}}{A}\left(erf\left(\frac{AD+B}{\sqrt{2}}\right) - erf\left(\frac{B}{\sqrt{2}}\right)\right) \\ &= -P\cdot\left(erf\left(\frac{AD+B}{\sqrt{2}}\right) - erf\left(\frac{B}{\sqrt{2}}\right)\right), \end{aligned} \tag{11}$$

where $erf(z) = \int_0^z e^{-z^2}dx$ and $P = \alpha^j_{\max}\sqrt{\frac{\pi}{2}}\frac{e^{-\frac{C}{2}}}{A}$. When A is less than or equal to zero, R_j can be evaluated using (7). Thus, the final opacity is $\alpha = 1 - \exp(\sum_{j=1}^n R_j)$.

Previous work [16], [28] approximates the erf function using a 2D texture, requiring numerical integration. Instead, we analytically evaluate it using a GPU, leading to high-quality preintegrated volume rendering. Figs. 3a and 3c compare our result with that of numerically integrated rendering at the same sampling rate. Fig. 3b shows the result produced by a numerical integration scheme with a doubled sampling rate. In terms of achieving comparable visual quality (Figs. 3a and 3b), our approach achieves better performance than the numerical integration approach.

3.4 Elliptical Classification Widgets

Unlike the inverse triangular or rectangular widgets used in the previous work [16], [28], the manipulation primitives in our approach are arbitrarily directional elliptical primitives. Moreover, the operations can be represented as a variety of transformations. The center of the elliptical primitive is the mean value μ. The other parameters can be computed by applying singular value decomposition [22] to the matrix Σ^{-1}

$$\begin{aligned} \Sigma^{-1} &= \begin{bmatrix} a & b \\ c & d \end{bmatrix} \\ &= \begin{bmatrix} -\cos(\phi) & \sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix}\begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix}\begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix}^T, \end{aligned} \tag{12}$$

where $\frac{1}{\sqrt{\sigma_1}}$ and $\frac{1}{\sqrt{\sigma_2}}$ are the radii along the major and minor axes and ϕ and ψ are the two angles that rotate the coordinate axes to the major and minor axes, respectively. For a symmetric 2×2 matrix, ϕ is equal to ψ.

After obtaining the parameters of the elliptical primitive, the following affine transformations can be applied:

- **Translation**—shifting the mean μ. The user can move the widget in feature space to explore features of interest. This transformation is guided by the user's domain knowledge about the feature space. For example, moving the widget toward a higher gradient magnitude region in the density and density gradient feature space can enhance feature boundaries.
- **Scaling**—scaling the radii of the principal axes σ_1 and σ_2. In our experience, the scaling operation is often guided by observing the extent of the corresponding feature. The initial ETFs usually overlap (Fig. 1a), and therefore appropriate scaling can improve feature separation. However, the user should be careful to avoid missing important structures, or introducing undesired or distracting features.
- **Rotation**—rotating the elliptical widget. This adds an angle β to ϕ and ψ in (12), leading to a new covariance matrix Σ. As the direction of the ETF characterizes the feature distribution [11], choosing an appropriate direction can improve the accuracy of feature identification. In our experience, such a direction can be found in several attempts by observing changes in the rendered image and the shape of the histogram.

  Fig. 4 illustrates the four operations on the ETF in red, as shown in Fig. 1a: recoloring, translating, scaling, and rotating. By interactively specifying each mixture component, the corresponding volumetric structures can be observed, providing a context to modulate the transfer function.
- **Subdivision.** Some mixture components may contain more than one feature, as illustrated by the ETF in dark red in Fig. 2a. To find more interesting small-scale features in an ETF, two operations can be performed. First, the greedy EM algorithm can be

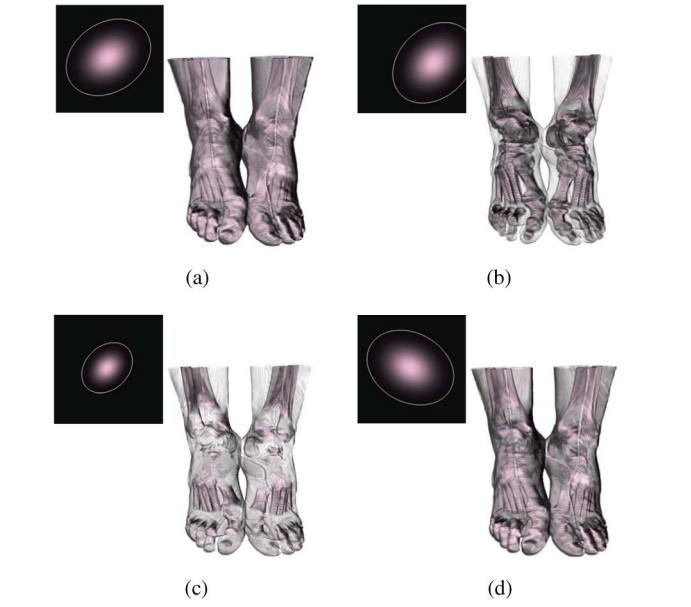


Fig. 4. Manipulating the ETF in red shown in Fig. 1a. (a) Recoloring from dark red to pink. (b) Translating the recolored ETF in (a) 0.21 along the x-axis and -0.15 along the y-axis. (c) Scaling the recolored ETF in (a) by a factor of 0.38 along both the major and minor axes. (d) Rotating the recolored ETF in (a) 45 degrees counter-clockwise.

employed again to yield more ETFs (Fig. 2). However, the adjustments made to the original ETFs cannot be incorporated into the subdivision process. This is a limitation of our current EM algorithm. An alternative solution is to subdivide an ETF directly into two pieces along the major axis, yielding two half radii for σ_1 and σ_2. The user can also use the scaling operation to refine them when this subdivision obscures interesting features. Fig. 5 shows an example where the joints of the feet can be distinguished from other structures using the subdivision operation. Clearly, we should not expect the subdivision to always produce a better classification. If the classification results are unsatisfactory, the user can easily backtrack to the original ETF.

4 Coherent Exploration of Time-Varying Data

In this section, we describe our GMM-based volume classification scheme for exploring time-varying data sets. In time-varying data sets, features of interest may evolve dynamically. Sudden appearances and disappearances are common phenomena. This may lead to sharp variations in the feature space, making coherent visualization of the entire time sequence a difficult task. However, these phenomena usually only occur in several keyframes. Consequently we divide the sequence into a list of time intervals in which the number of interesting features is fixed. The subdivision can be accomplished by utilizing the user's prior knowledge and/or automatic keyframe detection. Automatically detecting these sharp variations is beyond the scope of this paper. For details we refer to Wang et al. [36] and Lee and Shen [17]. Domain scientists usually have adequate knowledge and experience regarding when features of interest are likely to appear. In our work, the time intervals are manually constructed. In the rest of this section, we focus on achieving coherent user-guided exploration of one given time interval.

Fig. 5. Subdividing an ETF into two smaller ones. (a) The original ETF and the rendering result. (b) The two subdivided and recolored ETFs and the associated result, where the joints are differentiated from other structures.

There are two ways to achieve this goal. A straightforward way is to construct the GMM individually for each time step, which we refer to as *individual* classification. Although this does not incur any additional cost, its classification results cannot maintain temporal coherence without considering other time steps. Another way is to construct a volume of the feature space with one axis representing time and the others representing the feature space dimensions; this volume is then clustered with the GMM. However, constructing and clustering this feature space volume requires a large amount of storage and time. Guided by the principle of data stream clustering "to maintain a consistently good clustering of the sequence observed so far, using a small amount of memory and time" [10], we introduce a coherent classification scheme with the help of the incremental GMM estimation algorithm [3]. We call this method *incremental* classification.

We call an ETF computed from the incremental classification process a *suggestive* ETF, which may be adjusted later by the user. However, manually adjusting the ETFs for all time steps imposes a heavy workload on the user and may lead to temporally incoherent results. To resolve this issue, we allow the user to select a time step for making adjustments. We then automatically transfer these adjustments to the suggestive ETFs of other time steps. One straightforward way is to apply these adjustments directly to other time steps, which may yield good results for data with a small shift in histograms over time. We call this approach *direct transfer*. However, because most time-varying data sets are dynamic in nature, direct transfer may result in temporal incoherence. Accordingly, we propose a coherent propagation scheme that considers the evolution of feature space. We call this method *coherent propagation*.

4.1 Incremental Classification

We first define some notation. For the tth time step, the vectors in the feature space are $\{x_t^1, \ldots, x_t^n\}$, and the parameter set is $\theta_t = \{\alpha_t^j, \mu_t^j, \Sigma_t^j\}_{j=1}^k$, where k is the number of Gaussian components, and E_j^t is the cumulated posterior probability of the jth Gaussian.

An initial GMM with a parameter set θ_t is created by the EM algorithm for the tth time step. The tth time step is either the first step or a specified step. The vectors of the

$t+1$th time step and θ_t are the input to the incremental GMM estimation algorithm.

The incremental GMM estimation algorithm separates the data into two parts: one is dedicated to the data already used to train θ_t and the other to the data at the $t+1$th time step. The differences between adjacent time steps tend to be small; thus, we can assume that the set of the posterior probability $\{P(j|x_t^i)\}_{i=1}^n$ remains the same when the new data set $\{x_{t+1}^1, \ldots, x_{t+1}^n\}$ updates this classification. Thus, the cumulative posterior probability E_t^j of each Gaussian for the feature of the tth time step remains unchanged during the update. To maximize the likelihood $\prod_{i=1}^n p(x_{t+1}^i|\theta_{t+1})$, the EM procedure can be rewritten as follows:

- E Step:

$$\begin{aligned} P(j|x_{t+1}^i) &= \frac{\widehat{\alpha}_{t+1}^j g(x_{t+1}^i|\widehat{\mu}_{t+1}^j, \widehat{\Sigma}_{t+1}^j)}{\sum_{j=1}^k \widehat{\alpha}_{t+1}^j g(x^i|\widehat{\mu}_{t+1}^j, \widehat{\Sigma}_{t+1}^j)}, \\ E_{t+1}^j &= \sum_{i=1}^n P(j|x_{t+1}^i). \end{aligned} \tag{13}$$

- M Step:

$$\begin{aligned} \widehat{\alpha}_{t+1}^j &= \frac{E_t^j + E_{t+1}^j}{2n}, \\ \widehat{\mu}_{t+1}^j &= \frac{E_t^j \mu_t^j + \sum_{i=1}^n P(j|x_{t+1}^i) x_{t+1}^i}{E_t^j + E_{t+1}^j}, \\ \widehat{\Sigma}_{t+1}^j &= \frac{\sum_{i=1}^n P(j|x_t^i)(x_t^i - \widehat{\mu}_{t+1}^j)(x_t^i - \widehat{\mu}_{t+1}^j)^T}{E_t^j + E_{t+1}^j} \\ &\quad + \frac{\sum_{i=1}^n P(j|x_{t+1}^i)(x_{t+1}^i - \widehat{\mu}_{t+1}^j)(x^i - \widehat{\mu}_{t+1}^j)^T}{E_t^j + E_{t+1}^j} \\ &= \frac{E_j^t\left(\Sigma_t^j + (\mu_t^j - \widehat{\mu}_{t+1}^j)(\mu_t^j - \widehat{\mu}_j^{t+1})^T\right)}{E_t^j + E_{t+1}^j} \\ &\quad + \frac{\sum_{i=1}^n P(j|x_{t+1}^i)(x_{t+1}^i - \widehat{\mu}_{t+1}^j)(x^i - \widehat{\mu}_{t+1}^j)^T}{E_t^j + E_{t+1}^j}, \end{aligned} \tag{14}$$

where the variables with a hat will be iteratively updated until some convergence criterion is met. Note that θ_t and E_t remain the same in the classification of the $t+1$th time step. Compared with (6), (14) updates $\widehat{\alpha}_{t+1}^j$, $\widehat{\mu}_{t+1}^j$, and $\widehat{\Sigma}_{t+1}^j$ by taking the new incoming data and the estimated GMM parameters of the previous time steps as an entity. Accordingly, the additional memory requirement is $O(k)$, where k is the number of the mixture components. By setting θ_t as the initial parameters, convergence can be quickly achieved. As the correspondence between the feature and the Gaussian remains unchanged, the user can globally set the color and opacity for each feature from all the time steps. To demonstrate the effectiveness and robustness of incremental classification, we created a synthetic time-varying data set ($128 \times 128 \times 128 \times 40$) in which several concentric spheres move together. In this data set, the scalar values in the regions bounded by these spheres vary with time. To explore the movements of these spheres, we start incremental classification from the first time step, whose individual classification result is shown in Fig. 6a. Fig. 6b shows the incremental result of the 30th time step where four spheres are clearly shown. Although applying the individual classification of the 30th time step can characterize some features, as shown in Fig. 6c, it captures the wrong boundary for the red sphere and misses the inner sphere.

Notice that in the 30th time step, the right arc in the histogram is small and individual classification will treat it as one feature. This will not occur for the first time step because the right arc in its histogram is much larger. In contrast, incremental classification captures small variations of the histogram for every time step and produces coherent classification results.

4.2 Coherent Adjustment Propagation

With the classification results provided by the incremental EM algorithm, user adjustments are usually indispensable because they reflect domain knowledge or preference. Incremental classifications catch the small differences between adjacent time steps by updating the parameters of the mixture components; hence, user adjustments of these parameters should be updated accordingly for each time step. To achieve temporal coherence, we design a coherent propagation technique to transfer user adjustments of a selected time step automatically to other time steps.

Based on the assumption that the mixture components capture the variations of features, we first find affine transformations in the feature space that match the mixture components pairwise between two counterparts. These transformations are then used to depict the correspondence of histograms between the two time steps and are later applied to propagate the adjustments. To keep the number of Gaussians fixed, user adjustments are limited to affine transformations, as mentioned in Section 3.4.

For a feature, the Gaussian components at the current and next steps are

$$\begin{aligned} G(x_t;\theta_t) &= e^{(x_t-\mu_t)^T\Sigma_t^{-1}(x_t-\mu_t)}, \\ G(x_{t+1};\theta_{t+1}) &= e^{(x_{t+1}-\mu_{t+1})^T\Sigma_{t+1}^{-1}(x_{t+1}-\mu_{t+1})}, \end{aligned}$$

respectively, where x_t and x_{t+1} are the vectors of the tth and $t+1$th time steps, respectively. Similarly, we denote the adjusted and to-be-adjusted Gaussians as

$$\begin{aligned} \widetilde{G}(x_t;\widetilde{\theta}_t) &= e^{(x_t-\widetilde{\mu_t})^T\widetilde{\Sigma_t^{-1}}(x_t-\widetilde{\mu_t})}, \\ \widetilde{G}(x_{t+1};\widetilde{\theta}_{t+1}) &= e^{(x_{t+1}-\widetilde{\mu_{t+1}})^T\widetilde{\Sigma_{t+1}^{-1}}(x_{t+1}-\widetilde{\mu_{t+1}})}. \end{aligned}$$

An affine transformation T in the feature space is

$$x_{t+1} = Tx_t = Ax_t + d, \tag{15}$$

which transforms x_t to x_{t+1}. It can be determined by matching the components of these two time steps

$$G(x_t;\theta_t) = G(Ax_t + d;\theta_{t+1}),$$

which leads to

$$\begin{aligned} A^T\Sigma_{t+1}^{-1}A &= \Sigma_t^{-1}, \\ A^{-1}(\mu_{t+1} - d) &= \mu_t. \end{aligned} \tag{16}$$

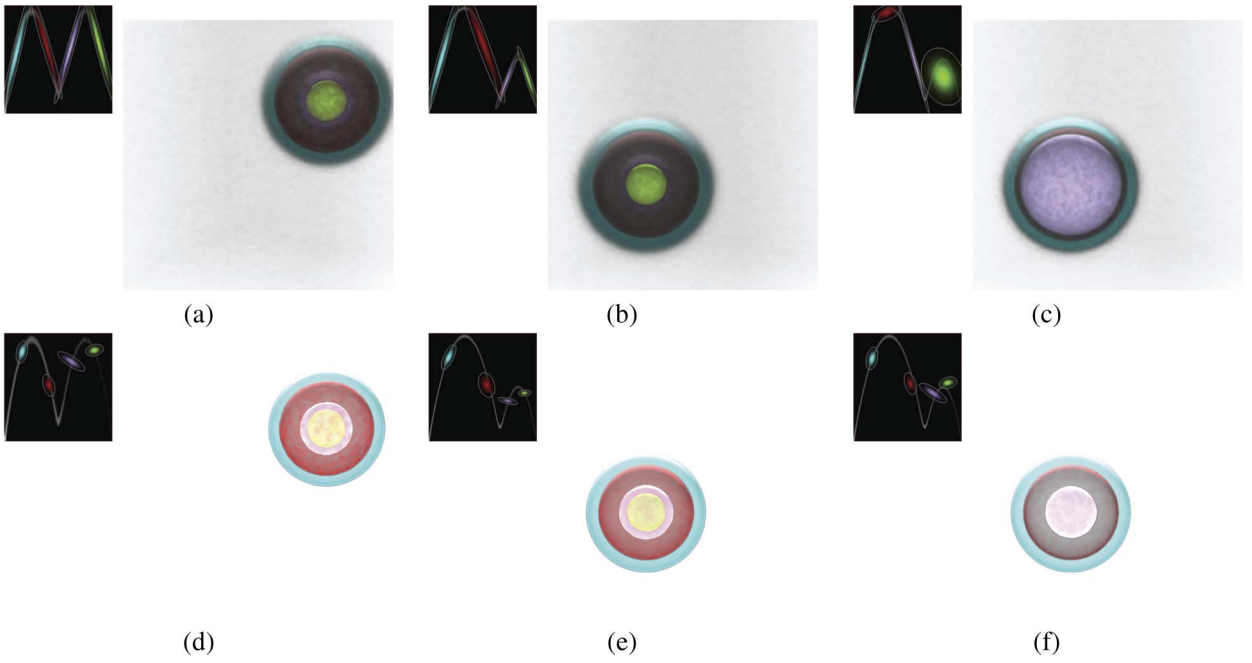


Fig. 6. Illustration of coherent propagation and direct transfer with a synthetic time-varying data set. (a) Classification result of the first time step produced by individual classification. (b) Incremental classification result of the 30th time step based on (a). (c) Individual classification result of the 30th time step. (d) Result produced by making adjustments to the first time step. (e) Adjusted result of the 30th time step by coherent propagation with respect to the adjustments from (a) to (d). (f) Classification result of the 30th time step produced by direct transfer with respect to the adjustments from (a) to (d).

The transformation T and the adjusted $\widetilde{G}(x_t;\widetilde{\theta}_t)$ are then used to determine the parameter $\widetilde{\theta}_{t+1}$ by matching

$$\widetilde{G}_t(x_t) = \widetilde{G}_{t+1}(Tx_t).$$

Applying the Cholesky decomposition [22], the parameters $\widetilde{\Sigma}_{t+1}$ and $\widetilde{\mu}_{t+1}$ can be easily found as follows:

$$\begin{aligned} A &= L_{t+1}L_t^{-1}, \\ \widetilde{\Sigma}_{t+1} &= A\widetilde{\Sigma}_t A^T, \\ \widetilde{\mu}_{t+1} &= \mu_{t+1} + A(\widetilde{\mu}_t - \mu_t), \end{aligned} \tag{17}$$

where $\Sigma_{t+1} = L_{t+1}L_{t+1}^T$ and $\Sigma_t = L_tL_t^T$. As such, other suggestive ETFs of the $t+1$th time step and the transfer function of its next time step can be automatically updated.

Compared with the results produced by the direct transfer, the coherent propagation considers the differences between two histograms. Figs. 6e and 6f show a comparison of the classification results produced by these two methods. This example shows the advantage of coherent propagation over direct transfer. The latter ignores differences between the suggestive ETFs at different time steps, making the adjustments on them miss the modification that corresponds to the evolution of the features. In contrast, coherent propagation considers the evolution of features captured by the incremental EM algorithm and modifies the adjustment accordingly. From the results, we can see that coherent propagation is more appropriate in adjusting the positions, sizes, and orientations of the suggestive ETFs.

Suppose the user wants to remove noise around the spheres (Fig. 6a). He/she then adjusts the suggestive ETFs of the first time step until the desired result is achieved, as shown in Fig. 6d. As there is a one-to-one correspondence between the features in different time steps, coherent propagation can transfer the user adjustments to the 30th time step. Fig. 6e shows the propagation result where four spheres are visible and become more clear. However, the red sphere becomes thinner and the green sphere is missed in Fig. 6f, which is caused by directly applying the user adjustments to Fig. 6b. A comparison of these two kinds of time-series results is shown in the supplemental video, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2011.97.

So far, we have only described a forward propagation workflow, where the user adjusts a selected time step and propagates the adjustments forward. Incremental classification allows for coherent backward propagation by deriving the GMM of the $t-1$th time step from the tth time step. Fig. 11 shows an example.

5 Implementation Details

We implemented and tested our approach on a PC with an Intel Core 2 Duo E6320 1.8 GHZ CPU, 2.0 GB RAM, and an NVIDIA Geforce GTX 260 video card (512 MB video memory), using the Cg Language. All images shown in this paper were generated at a resolution of $1{,}024 \times 768$. We describe the implementation details in the context of exploring one data set because it is almost the same for a sequence of data sets.

The core part of our scheme is the greedy EM algorithm. We use accelerated greedy EM [34], which has proven to be convergent for large data sets. It first organizes data into a *kd*-tree structure to precompute the statistical variables used in the optimization. The algorithm then uses the partitioned blocks to perform the optimization.

There are two methods for using greedy EM clustering. The first one performs EM clustering only once throughout

TABLE 1
Clustering and Rendering Performances for the Engine Data Set ($256 \times 256 \times 256$)

#ETFs	One-round	Multi-round	Rendering
3	413ms	413ms	56fps
4	795ms	675ms	44fps
5	1301ms	948ms	29fps
6	2289ms	1241ms	23fps

TABLE 2
Rendering Performances for Six Data Sets

Data	Size	#ETFs	Performance
Engine	$256 \times 256 \times 128$	4	44fps
Feet	$256 \times 256 \times 256$	4	39fps
Horseshoe Vortex	$256 \times 269 \times 256$	3	41fps
Carp	$256 \times 256 \times 512$	3	38fps
Head	$512 \times 512 \times 381$	3	29fps
Hurricane	$500 \times 500 \times 100$	3	41fps

the entire procedure. After the desired number of components is generated in the first stage, these components can be freely manipulated. We call this the *one-round* clustering. In contrast, the second method is a *multiround* procedure in which new components can be inserted progressively based on an arbitrary initial separation. The *multiround* procedure is preferred in our approach because it achieves both high performance and sufficient flexibility. The second and third columns of Table 1 compare the performance of these two methods.

Rendering performance gradually decreases as the number of mixture components increases, because ETF-based preintegrated volume rendering is directly evaluated on the GPU. As listed in the fourth column of Table 1, doubling the number of ETFs makes rendering performance drop in half. With a fixed number of components, rendering performance decreases as the data size increases. In Table 2, we can see that our GMM-based volume renderer can achieve interactive frame rates for medium-sized volume data.

As shown in Tables 1 and 2, the number of ETFs has the most significant influence on performance. In our experiments, we found that a small number of mixture components are sufficient for good approximations of the distribution of 2D feature space, while offering a rich space for exploration. As shown in Table 1, generation of five mixture components typically takes less than one second with an unoptimized CPU implementation.

6 Applications

When provided with a 2D histogram, the user may have few ideas regarding which samples in the feature space correspond to meaningful structures in the volumetric data set. Based on maximum likelihood estimation, our approach automatically decomposes the feature space into several regions that denote meaningful structures. If the initial result is not satisfactory, the user can iteratively tune the suggestive ETFs, to better understand further the relationship between volumetric structures.

The GMM-based exploration scheme can be applied to many kinds of meaningful feature spaces, because it is independent of the definition of the feature space. In addition to the widely used density and density gradient magnitude, other meaningful variables can be incorporated into the feature spaces. By applying the GMM to these feature spaces, the user is equipped with an exploration tool for feature classification, knowledge-aware multivariate volume exploration, and temporally coherent transfer function design.

6.1 Arc-Shaped Feature Space

To demonstrate the effectiveness of our two-stage exploration scheme on the density and density gradient magnitude feature space, we used the Feet data set. The first clustering step produces four ETFs, as shown in Fig. 1a, where the ETFs in red and plum dominate. The ETF in red corresponds to the skin and phalanges, whereas the ETF in plum corresponds to a portion of the skin and the ankle. Most of the voxels in the data set belong to these two ETFs. However, the skin identified by the ETF in red occludes the phalanges, and parts of the skin identified by the ETF in plum occludes the ankle. These two ETFs overlap with each other, and both identify a portion of the skin. Thus, we first scaled them to improve the separation between the phalanges and the ankles. Afterward, we rotated the ETF in red to a better direction. After these manipulations, a more desirable result was obtained, as shown in Fig. 1b, where these two parts become clearly differentiated. From the experiment, we can see that our two-stage exploration scheme is capable of quickly obtaining the desired classification results.

To demonstrate the effectiveness of our exploration scheme in reconstructive surgery, we conducted another experiment on a CT facial deformity data set, as shown in Fig. 7. The data set was acquired from a patient suffering from a facial deformity. The damaged regions located near the upper jaw and the top of the skull must be identified in the surgical planning procedure. We obtained the initial result after clustering the 2D histogram with three mixture components, as shown in Fig. 7b, where two regions are vaguely shown: a lesion in cyan and a damaged region where some teeth are absent. However, the relationship between these regions and the bones of the head and face is not clear. We noticed that the ETF in gray, corresponding to the skin, overlaps with the others located in the low gradient magnitude region. We shrank it to achieve better feature separation. To enhance its boundary, we then moved it to a higher gradient magnitude region and rotated it to align with the direction of the nearby arc. We handled the other features in a similar fashion. After these adjustments, a better result (Fig. 7c) was achieved, where the lesion (marked in cyan) and the damaged region where some teeth are absent are clearly illustrated. From this result, we can see that parts of the teeth left to the nose are lost and that the face is deformed toward the right. To investigate the lesion located at the top of the skull, the user explores the data set from another viewpoint and finds a large crack, as shown in Fig. 7d. From this experiment, we can see that our GMM exploration scheme provides an effective navigation interface for the user to explore the relationships between structures freely.

6.2 Arbitrarily Shaped Feature Space

The above two examples involve medical data sets. The histograms of the density and density gradient magnitude of these data sets exhibit arc shapes, representing material

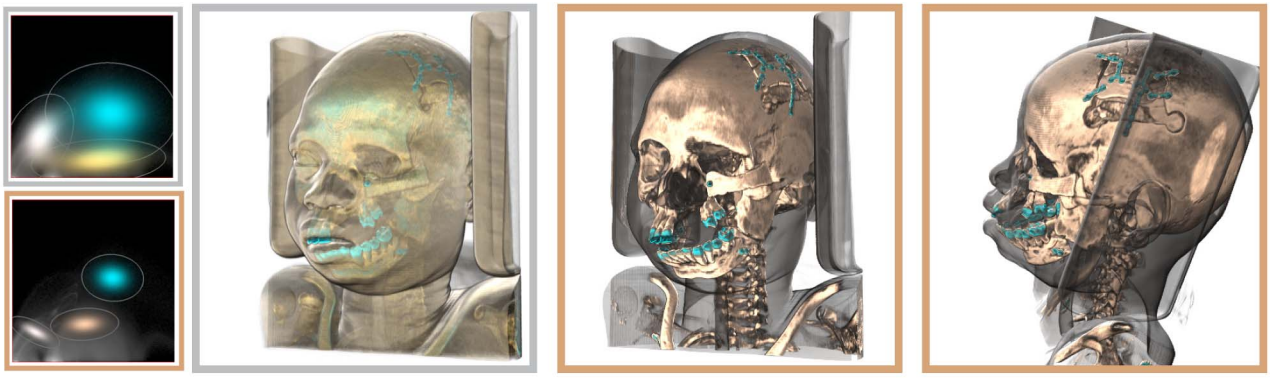


Fig. 7. Exploring the CT facial deformity data set using the density and density gradient magnitude feature space. The three automatically generated ETFs (first row of (a)) produce a result (b) that vaguely depicts the skin and the damaged region on the head. After adjusting these three ETFs (second row of (a)), the damaged regions become clearly distinguished (c, d).

boundaries. With this intuition, the user simply places the ETFs along the arcs and obtains a reasonably good image. However, specifying the ETFs for many scientific data sets can be quite challenging because they do not have clear boundaries, and thus their histograms do not exhibit arc shapes. In contrast, the GMM-based scheme works well by supporting a continuous, probabilistic representation, whether or not the data set has an arc-shaped distribution.

To illustrate the advantage of our approach in the exploration of data sets without a discernable arc shape, our next experiment utilized the Horseshoe Vortex data set shown in Fig. 8. Here, the feature space consists of the second invariant of the velocity gradient and its gradient magnitude. We can see the vortex tubes, their intersections, and some distracting noise from the initial classification result (middle, Fig. 8). To show these structures more clearly, the user scales and moves the ETFs, obtaining a better result (right, Fig. 8).

6.3 Knowledge-Aware Feature Space

Other variables can be employed in addition to the feature space composed of density and density gradient magnitude. Usually, a user who works on multivariate time-varying data has specific domain knowledge for constructing a suitable feature space. Representing these kinds of feature spaces with GMM-enabled probabilistic representations can favorably characterize feature separation, facilitating quick discovery of particularly interesting features.

To demonstrate the effectiveness of our scheme in multivariate feature space, we conducted an experiment on the Turbulent data set (Fig. 9) produced by a 128-cubed simulation of a compressible, turbulent slip surface. Vortices exist in the region with a large vorticity and small pressure. Thus, we applied the GMM exploration scheme to the vorticity versus the pressure feature space and obtained four mixture components (top left, Fig. 9). Among these four ETFs, the user is not interested in the ones located in the regions with small vorticities and large pressures. After removing these two ETFs and adjusting the other two ETFs, a better result (right, Fig. 9) is obtained, which reveals the kinking and tangling vortex tubes.

Note that the feature space in this example does not exhibit any arc-like shape. Our approach still yields

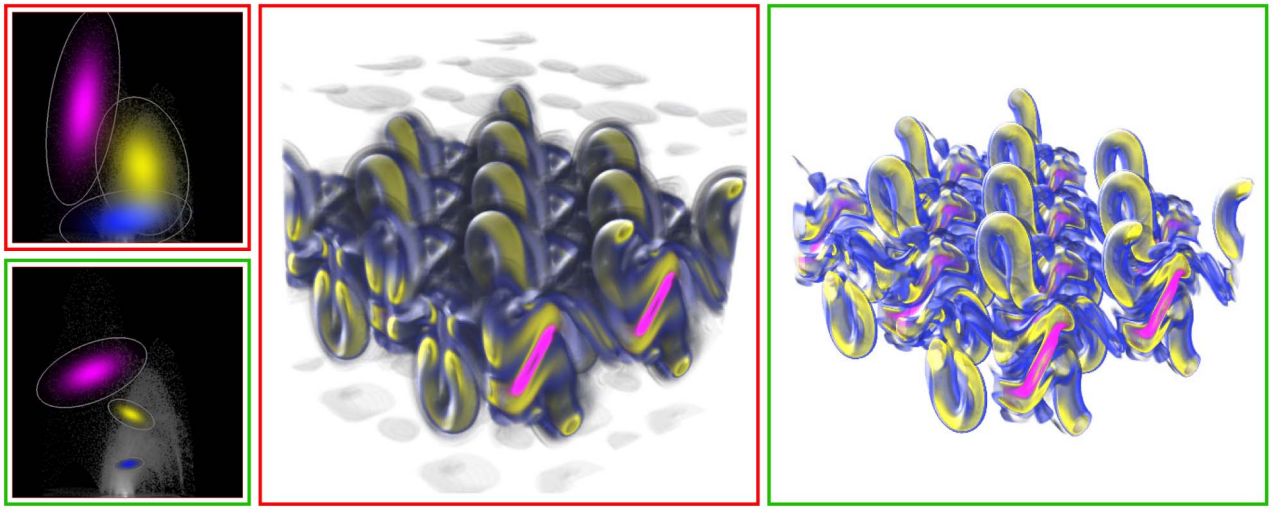


Fig. 8. Exploring the Horseshoe Vortex data set using the second invariant of the velocity gradient and its gradient magnitude feature space. The three automatically generated ETFs (top left) produce a result (middle) that shows some noise. By manipulating these three ETFs (bottom left), the interior vortex tubes are clearly shown (right).

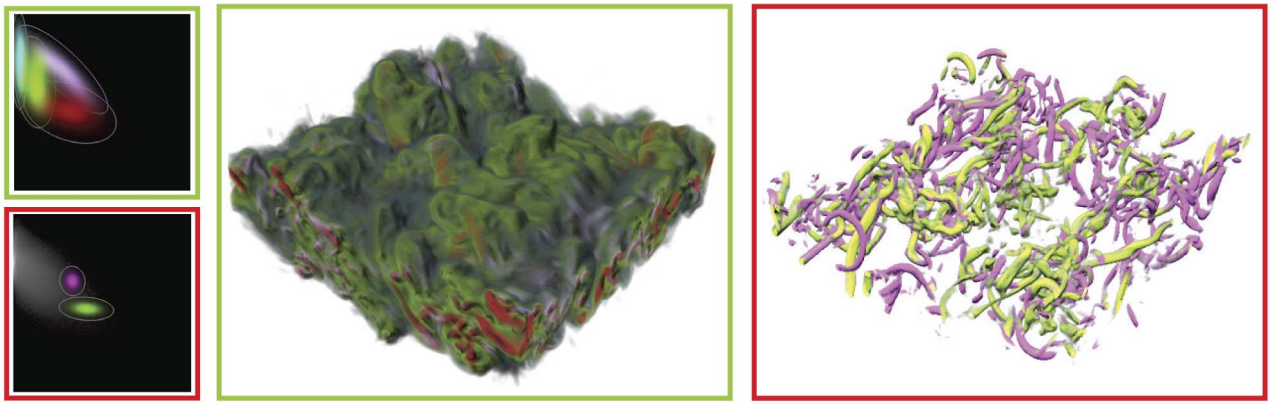


Fig. 9. Exploring the Turbulent data set in the vorticity and pressure feature space. The result (middle) with the automatically generated four ETFs (top left) does not clearly show the vortex tubes. By performing a sequence of operations, namely, removing the two ETFs located in the regions with small vorticities and large pressures as well as scaling and recoloring the other two ETFs, the kinking and tangling vortex tubes become clearly shown with a large contrast.

acceptable results by clustering the histogram space into varied regions that correspond to different ranges of vorticity and pressure. Moreover, the user can adjust the suggestive results according to his/her domain knowledge.

6.4 Time-Varying Data

In time-varying data visualization, coherence plays an important role in correctly interpreting the rendered images. After adjusting the initial ETFs of a selected time step, the evolution of features of interest can be easily identified from the semiautomatically generated visualization results. To demonstrate the effectiveness of our approach, we applied it to two different time-varying data sets. The time-series animations of these two data sets generated by our method are included in the supplemental electronic material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2011.97.

In the first case study, we used the Vortex data set ($128 \times 128 \times 128 \times 100$) generated from a pseudospectral simulation of vortex structures [27]. The suggestive classification results show three regions corresponding to low vorticity, mid vorticity with low gradient, and high vorticity with high gradient. To make the tubes clearer, we performed several manipulations on the initial classification result of the first time step (first row in Fig. 10a). In this result, three layers of vortex tubes with different ranges of vorticity magnitude are clearly shown (second row of Fig. 10a). These adjustments are then propagated to the other 99 time steps, producing coherent results where the vortex structures gradually become larger. The second row in Figs. 10b, 10c, and 10d shows the results of the 19th, 69th, and 94th time steps, respectively. The first row in Figs. 10b, 10c, and 10d shows the classification results of these time steps generated by the direct transfer. The vorticity magnitude in the simulation is a continuous function; thus, the depth order of the low, mid, and high vorticity features should be preserved during the evolution. Comparing these two groups of classification results, we can see that the exterior vortex tubes are always maintained in the propagation-based results, whereas the result produced by the direct transfer misses the exterior layer, as shown in Figs. 10b and 10d. Moreover, the result in the first row of Fig. 10c misses the interior high vorticity tube because the sizes and relative locations of the corresponding Gaussian components are incoherent with those in the user adjusted time step. From this experiment, we can see that the

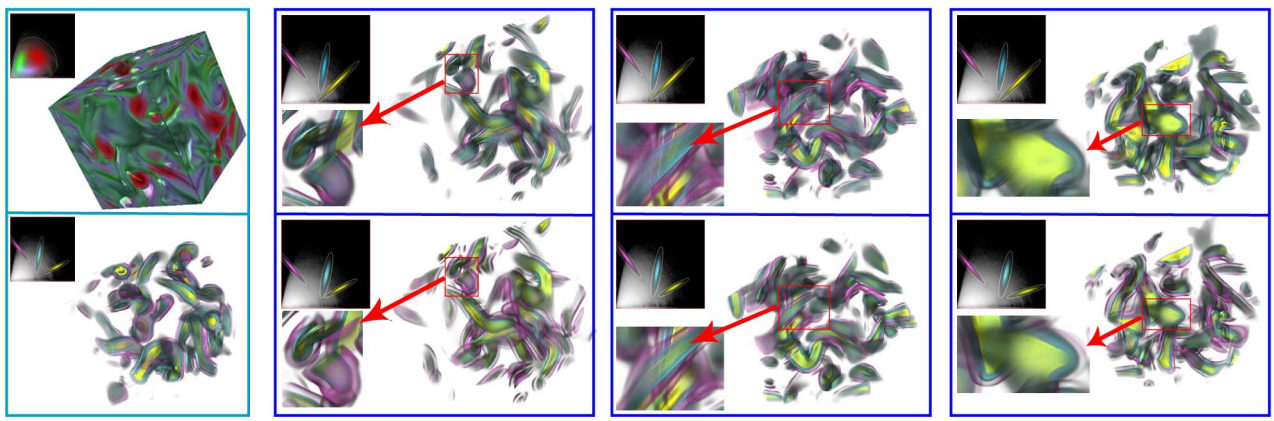


Fig. 10. Coherent exploration of the Vortex data set. (a) The initial classification result of the first time step (first row) and the adjusted result containing a three-layered vortex tube (second row). The first row of (b, c, d) shows the classification results of the 19th, 69th, and 94th time steps by directly transferring the ETF adjustments in (a). Using our coherent adjustment propagation method, coherent results that preserve the depth are generated (second row of (b, c, d)).

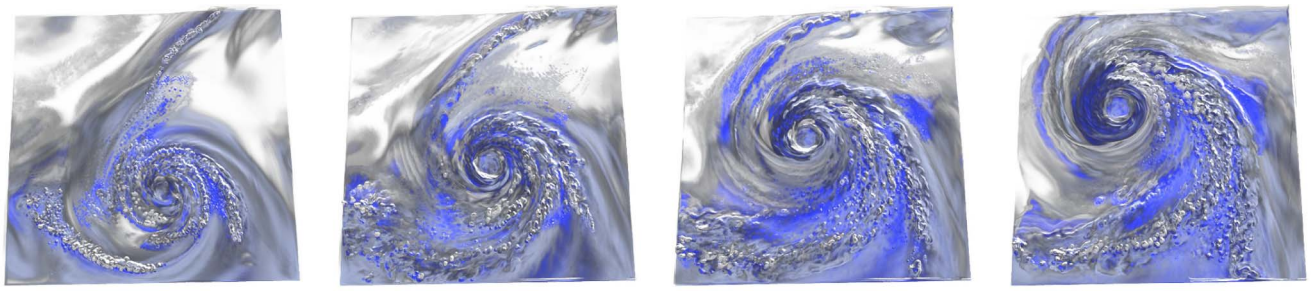


Fig. 11. Coherent exploration of the *vapor* variable in the Hurricane *Isabel* data set. After adjusting the initial ETFs of the 30th time step, the regions with high, middle, and low vapor are indicated in blue, skyblue, and gray, respectively, as shown in (c). These adjustments are then backward and forward propagated to all the other time steps. (a, b, d) Classification results of the 10th, 20th, and 40th time steps, respectively.

proposed coherent propagation generates more consistent adjustments to the ETFs. Moreover, the resulting images are more coherent in feature evolution.

Incremental propagation also works for backward propagation. We tested two-way propagation using the data set Hurricane *Isabel*, the benchmark for the IEEE 2004 Visualization Design Contest. This data set was generated by the National Center for Atmospheric Research to simulate a strong hurricane in the west Atlantic region in September 2003. It has 48 time steps, with a resolution of $500 \times 500 \times 100$. In this data set, the researchers are concerned with how the variables relate to the evolution of the hurricane eye. We studied the water vapor mixing ratio (*vapor*). After dividing *vapor* into three ranges, i.e., high, middle, and low *vapor*, we adjusted the initial classification of the 30th time step and obtained the result, shown in Fig. 11c. In this adjusted result, the high vapor region is indicated in blue, the middle in sky blue, and the lower in gray. We can see that the region close to the hurricane eye has high *vapor*. This is reasonable because *vapor* is the fuel of a hurricane. To find out how *vapor* evolves with the hurricane eye, the adjustments for the 30th time step are forward and backward propagated to other steps. Figs. 11a, 11b, and 11d show the classification results of the 10th, 20th, and 40th time steps. From these four time steps, we can see that the *vapor* gradually increases and that there is an increasing number of bubbles in the trajectory of the hurricane eye.

6.5 Limitations

Although our approach provides an easy-to-use exploration mechanism, it is still a semiautomatic method and requires some user supervision. First, the number of the mixture components needs to be provided, because the user may not have a clear idea about how many features of interest are in the data set. A recently developed Bayesian variation framework for the mixture model [4] can solve this problem. It simultaneously estimates the number of mixture components and learns the parameters of the mixture model. Second, the interaction of the elliptical classification widgets is a pure manipulation in the feature space. Although we provided some empirical guidelines in Section 3.4, we would like to integrate some guidance metrics for our scheme, e.g., visibility histograms [8].

Although incremental GMM estimation provides coherent classification, keeping a fixed number of Gaussians for all time steps may miss some features. In addition, the propagation is only valid within a specified time interval. To resolve these issues, we plan to investigate several solutions to divide time sequences into lists of intervals, with each interval having a fixed number of features. One possible solution is to use methods from importance-driven time-varying data visualization [36], which can automatically select multiple representative time steps. Another possible solution is temporal trend identification [17], in which dynamic time warping captures temporally coherent features. Our current incremental propagation method only considers the adjustments of one time step. We intend to propagate user adjustments on multiple representative time steps in future work.

7 Conclusion

This paper introduces a new volume exploration scheme with a unique ability to capture the data characteristics, while still affording favorable user interactivity. This flexibility is especially helpful to inexperienced users because it can automatically provide a suggestive volume classification using a greedy EM algorithm and an incremental GMM estimation scheme. By allowing the user to interactively select precomputed clusters in the feature space, he/she can gain an initial understanding of the underlying data set. Moreover, each cluster can be manipulated using an elliptical classification widget. By using the GPU, ETF-enabled transfer functions can be seamlessly incorporated with preintegrated volume rendering. For time-varying data sets, the manipulation of a selected time step can be propagated to other steps, yielding coherent classification.

Acknowledgments

The authors would like to thank Michael Knox for proofreading our paper, Jun Liu for helpful discussions in creating Fig. 1, and the anonymous reviewers for their valuable comments. This paper was partially supported by the Knowledge Innovation Project of Chinese Academy of Sciences (No. KGGX1-YW-13, No. O815011103), 973 program of China (2010CB732504), National Natural Science Foundation of China (No. 60873123, No. 60873113), 863 project (No. 2009AA062700), and Zhejiang Natural Science Foundation (No. 1080618). The data sets are courtesy of General Electronic, Deborah Silver, David Porter, XinLing Li, and the OsiriX Foundation.

REFERENCES

[1] H. Akiba, N. Fout, and K.L. Ma, "Simultaneous Classification of Time-Varying Volume Data Based on the Time Histogram," *Proc. IEEE/Eurographics Symp. Visualization '06,* pp. 171-178, 2006.

[2] J.A. Bilmes, "A Gentle Tutorial of the EM Algorithm and Its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models," Technical Report ICSI-TR-97-02, Univ. of Berkeley, 1998.

[3] S. Calinon and A. Billard, "Incremental Learning of Gestures by Imitation in a Humanoid Robot," *Proc. ACM/IEEE Int'l Conf. Human-Robot Interaction '07,* pp. 255-262, 2007.

[4] C. Constantinopoulos, M.K. Titsias, and A. Likas, "Bayesian Feature and Model Selection for Gaussian Mixture Models," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 28, no. 6, pp. 1013-1018, June 2006.

[5] C. Correa, Y.H. Chan, and K.L. Ma, "A Framework for Uncertainty-Aware Visual Analytics," *Proc. IEEE Symp. Visual Analytics Science and Technology '09,* pp. 51-58, 2009.

[6] C. Correa and K.L. Ma, "Size-Based Transfer Functions: A New Volume Exploration Technique," *IEEE Trans. Visualization and Computer Graphics,* vol. 14, no. 6, pp. 1380-1387, Nov./Dec. 2008.

[7] C. Correa and K.L. Ma, "The Occlusion Spectrum for Volume Visualization and Classification," *IEEE Trans. Visualization and Computer Graphics,* vol. 15, no. 6, pp. 1465-1472, Nov./Dec. 2009.

[8] C.D. Correa and K.L. Ma, "Visibility Histograms and Visibility-Driven Transfer Functions," *To Appear in IEEE Trans. Visualization and Computer Graphics,* 2011.

[9] I. Fujishiro, T. Azuma, and Y. Takeshima, "Automating Transfer Function Design for Comprehensible Volume Rendering Based on 3D Field Topology Analysis," *Proc. IEEE Visualization '99,* pp. 467-563, 1999.

[10] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan, "Clustering Data Streams," *Proc. Ann. Symp. Foundations of Computer Science (FOCS '00),* pp. 359-366, 2000.

[11] Y. Jang, R.P. Botchen, A. Lauser, D.S. Ebert, K.P. Gaither, and T. Ertl, "Enhancing the Interactive Visualization of Procedurally Encoded Multifield Data with Ellipsoidal Basis Functions," *Computer Graphics Forum,* vol. 25, no. 3, pp. 587-596, 2006.

[12] T. Jankun-Kelly and K.L. Ma, "A Study of Transfer Function Generation for Time-Varying Volume Data," *Proc. Eurographics/ IEEE VGTC Workshop Vol. Graphics '01,* pp. 51-68, 2001.

[13] G. Kindlmann and J.W. Durkin, "Semi-Automatic Generation of Transfer Functions for Direct Volume Rendering," *Proc. IEEE Symp. Vol. Visualization '98,* pp. 79-86, 1998.

[14] G. Kindlmann, R. Whitaker, T. Tasdizen, and T. Moller, "Curvature-Based Transfer Functions for Direct Volume Rendering: Methods and Applications," *Proc. IEEE Visualization '03,* pp. 513-520, 2003.

[15] J. Kniss, G. Kindlmann, and C. Hansen, "Interactive Volume Rendering Using Multi-Dimensional Transfer Functions and Direct Manipulation Widgets," *Proc. IEEE Visualization '01,* pp. 255-262, 2001.

[16] J. Kniss, S. Premoze, M. Ikits, A. Lefohn, C. Hansen, and E. Praun, "Gaussian Transfer Functions for Multi-Field Volume Visualization," *Proc. IEEE Visualization '03,* pp. 65-72, 2003.

[17] T.Y. Lee and H.W. Shen, "Visualization and Exploration of Temporal Trend Relationships in Multivariate Time-Varying Data," *IEEE Trans. Visualization and Computer Graphics,* vol. 15, no. 6, pp. 1359-1366, Nov./Dec. 2009.

[18] M. Levoy, "Display of Surfaces from Volume Data," *IEEE Computer Graphics and Applications,* vol. 8, no. 3, pp. 29-37, May 1988.

[19] R. Maciejewski, I. Wu, W. Chen, and D. Ebert, "Structuring Feature Space: A Non-Parametric Method for Volumetric Transfer Function Generation," *IEEE Trans. Visualization and Computer Graphics,* vol. 15, no. 6, pp. 1473-1480, Nov./Dec. 2009.

[20] N. Max, "Optical Models for Direct Volume Rendering," *IEEE Trans. Visualization and Computer Graphics,* vol. 1, no. 2, pp. 99-108, June 1995.

[21] H. Pfister, B. Lorensen, C. Bajaj, G. Kindlmann, W. Schroeder, L.S. Avila, K. Martin, R. Machiraju, and J. Lee, "The Transfer Function Bake-Off," *IEEE Computer Graphics and Applications,* vol. 21, no. 3, pp. 16-22, May/June 2001.

[22] W.H. Press et al., *Numerical Recipes.* Cambridge Univ. Pr., 1986.

[23] S. Roettger, M. Bauer, and M. Stamminger, "Spatialized Transfer Functions," *Proc. IEEE/Eurographics Symp. Visualization '05,* pp. 271-278, 2005.

[24] C.R. Salama, M. Keller, and P. Kohlmann, "High-Level User Interfaces for Transfer Function Design with Semantics," *IEEE Trans. Visualization and Computer Graphics,* vol. 12, no. 5, pp. 1021-1028, Sept./Oct. 2006.

[25] Y. Sato, C. Westin, A. Bhalerao, S. Nakajima, N. Shiraga, S. Tamura, and R. Kikinis, "Tissue Classification Based on 3D Local Intensity Structures for Volume Rendering," *IEEE Trans. Visualization and Computer Graphics,* vol. 6, no. 2, pp. 160-180, Apr.-June 2000.

[26] M.A. Selver and C. Güzelis, "Semi-Automatic Transfer Function Initialization for Abdominal Visualization Using Self-Generating Hierarchical Radial Basis Function Networks," *IEEE Trans. Visualization and Computer Graphics,* vol. 15, no. 3, pp. 395-409, May/June 2009.

[27] D. Silver and X. Wang, "Tracking and Visualizing Turbulent 3D Features," *IEEE Trans. Visualization and Computer Graphics,* vol. 3, no. 2, pp. 129-141, Apr.-June 1997.

[28] Y. Song, W. Chen, R. Maciejewski, K.P. Gaither, and D.S. Ebert, "Bivariate Transfer Functions on Unstructured Grids," *Computer Graphics Forum,* vol. 28, no. 3, pp. 783-790, 2009.

[29] C. Stauffer and W.E.L. Grimson, "Adaptive Background Mixture Models for Real-Time Tracking," *Proc. IEEE Conf. Computer Vision and Pattern Recognition '99,* pp. 246-252, 1999.

[30] A. Tikhonova, C. Correa, and K.L. Ma, "An Exploratory Technique for Coherent Visualization of Time-Varying Volume Data," *Computer Graphics Forum,* vol. 29, no. 3, pp. 783-792, 2010.

[31] F. Tzeng, E. Lum, and K. Ma, "An Intelligent System Approach to Higher-Dimensional Classification of Volume Data," *IEEE Trans. Visualization and Computer Graphics,* vol. 11, no. 3, pp. 273-284, May/June 2005.

[32] F. Tzeng and K. Ma, "A Cluster-Space Visual Interface for Arbitrary Dimensional Classification of Volume Data," *Proc. IEEE/Eurographics Symp. Visualization '04,* pp. 17-24, 2004.

[33] F. Tzeng and K. Ma, "Intelligent Feature Extraction and Tracking for Visualizing Large-Scale 4D Flow Simulations," *Proc. 2005 ACM/IEEE Conf. Supercomputing,* p. 6, 2005.

[34] J.J. Verbeek, J.R.J. Nunnink, and N. Vlassis, "Accelerated EM-Based Clustering of Large Data Sets," *Data Mining and Knowledge Discovery,* vol. 13, no. 3, pp. 291-307, 2006.

[35] J. Verbeek, N. Vlassis, and B. Krose, "Efficient Greedy Learning of Gaussian Mixture Models," *Neural Computation,* vol. 15, no. 2, pp. 469-485, 2003.

[36] C. Wang, H. Yu, and K.L. Ma, "Importance-Driven Time-Varying Data Visualization," *IEEE Trans. Visualization and Computer Graphics,* vol. 14, no. 6, pp. 1547-1554, Nov./Dec. 2008.

[37] Y.H. Wang, W. Chen, G.H. Shan, T.X. Dong, and X.B. Chi, "Volume Exploration Using Ellipsoidal Gaussian Transfer Functions," *Proc. IEEE Pacific Visualization Symp. '10,* pp. 25-32, 2010.

[38] J.L. Woodring and H.-W. Shen, "Semi-Automatic Time-Series Transfer Functions via Temporal Clustering and Sequencing," *Computer Graphics Forum,* vol. 28, no. 3, pp. 791-798, 2009.

[39] Y. Zhang, M. Brady, and S. Smith, "Segmentation of Brain MR Images through a Hidden Markov Random Field Model and the Expectation-Maximization Algorithm," *IEEE Trans. Medical Imaging,* vol. 20, no. 1, pp. 45-57, Jan. 2001.

[40] Z. Zhang, "Determining the Epipolar Geometry and Its Uncertainty: A Review," *Int'l J. Computer Vision,* vol. 27, no. 2, pp. 161-195, 1998.

[41] J. Zhou and M. Takatsuka, "Automatic Transfer Function Generation Using Contour Tree Controlled Residue Flow Model and Color Harmonics," *IEEE Trans. Visualization and Computer Graphics,* vol. 15, no. 6, pp. 1481-1488, Nov./Dec. 2009.

Yunhai Wang received the BEng degree in computer science from Southwest Normal University, PR China, in 2005. He is working toward the PhD degree in Supercomputing Center of Computer Network Information Center, Chinese Academy of Sciences (CAS). His research interests are in scientific visualization, visual analytics and high performance computing.

Wei Chen from June 2000 to June 2002, he was a joint PhD student in Fraunhofer Institute for Graphics, Darmstadt, Germany and received the PhD degree in July 2002. His PhD advisors were Professor Qunsheng Peng, and Professor Georgios Sakas. He is a professor in State Key Lab of CAD&CG at Zhejiang University, PR China. From July 2006 to September 2008, he was a visiting scholar at Purdue University, working in PURPL with Professor David S. Ebert. In December 2009, he was promoted as a full professor of Zhejiang University. He has performed research in computer graphics and visualization and published more than 60 peer-reviewed journal and conference papers in the last five years. His current research interests include scientific visualization, visual analytics, and bio-medical image computing. He is a member of the IEEE.

Jian Zhang received the BS degree in computational mathematics from Peking University, PR China in 1995, and the PhD degree in applied mathematics from the University of Minnesota in May 2005. He is a research scientist in Supercomputing Center of Computer Network Information Center, Chinese Academy of Sciences (CAS). From June 2005 to June 2009, he was a postdoc in the Pennsylvania State University working on scientific computing and modeling. His current research interests include scientific computing, high performance computing, and scientific visualization.

Tingxing Dong received the BS degree in physics from Zhengzhou University in 2007 and the master's degree in computer science from Graduate University of Chinese Academy of Sciences in 2010. He is working toward the PhD degree majoring in computer science in the University of Tennessee, Knoxville.

Guihua Shan received the BS degree in applied mathematics at JiShou University in 1997, the MS degree in computational mathematics at Hunan University in 2000, and the PhD degree in computer science at Computer Network Information Center, Chinese Academy of Sciences (CNIC, CAS) in 2010. She has been working in Supercomputing Center of CNIC, CAS since 2000. Her research interests include scientific visualization, information visualization, and image processing.

Xuebin Chi received the BS degree from Jilin University. He received the MS and PhD degrees from Computational Center of Chinese Academy of Sciences. He joined Software Institution, CAS in 1989. Due to his contribution to parallel computation, he won the second prize of the National Science and Technology Advancement Award of China in 2000. Currently he is the director and professor of Supercomputing Center of CNIC, CAS. His research interests include parallel computing and application, scientific visualization, and grid computing.