Scalable Multivariate Volume Visualization and Analysis based on Dimension Projection and Parallel Coordinates

Hanqi Guo, He Xiao, and Xiaoru Yuan, *Member, IEEE*

Abstract—In this paper, we present an effective and scalable system for multivariate volume data visualization and analysis with a novel transfer function interface design that tightly couples parallel coordinates plots (PCP) and MDS-based dimension projection plots. In our system, the PCP visualizes the data distribution of each variate (dimension) and the MDS plots project features. They are integrated seamlessly to provide flexible feature classification without context switching between different data presentations during the user interaction. The proposed interface enables users to identify relevant correlation clusters and assign optical properties with lassos, magic wand and other tools. Furthermore, direct sketching on the volume rendered images has been implemented to probe and edit features. With our system, users can interactively analyze multivariate volumetric data sets by navigating and exploring feature spaces in unified PCP and MDS plots. To further support large scale multivariate volume data visualization and analysis, Scalable Pivot MDS (SPMDS), parallel adaptive continuous PCP rendering, as well as parallel rendering techniques are developed and integrated into our visualization system. Our experiments show that the system is effective in multivariate volume data visualization and its performance is highly scalable for data sets with different sizes and number of variates.

Index Terms—Multivariate volume, transfer function, parallel coordinates, dimension projection, user interface design, parallel visualization.

1 Introduction

Multivariate volumetric data sets, which are comprised of many independent parameters that may or may not co-vary, are common in many applications. For example, in computational flow dynamics, velocity, pressure, temperature and other derived values such as vorticity are involved to define features. However, effectively visualizing and analyzing such multivariate data is still a challenging task. Existing methods of direct volume rendering have been successful with scalar field volume data, such as that produced in a CT scan. The original density values are mapped into colors and opacities through Transfer Functions (TF). Conversely, the mapping from multivariate volumetric data to visual components is much more difficult to design, because the high dimensional feature space is even harder to handle. The complexity of the problem is compounded as the scale of the multivariate data set becomes exceedingly large. As scientists attempt to solve larger and more complicated problems using massively parallel supercomputing power [40], multivariate datasets are produced at terabyte (TB) to petabyte (PB) scale. A key goal of multivariate volume rendering is to allow users to interactively interrogate the data with the goal of understanding the correlations between different variables with multivariate TF. A scalable system to effectively visualize multivariate TFs requires efficient parallel implementation.

In previous work [11], we developed a novel design of multivariate TFs, which tightly integrates two multidimensional visualization methods: PCP (Parallel Coordinates Plot) and MDS (Multi-Dimensional Scaling) plots. PCP [16] transforms points from high dimensional space to 2D space in the form of polylines and presents the information for individual dimensions. Furthermore, the technique simultaneously visualizes the correlations between dimension axes next to each other. Multi-Dimensional Scaling (MDS) techniques [39], are widely used to identify and select clusters. They project high dimensional points into a lower dimensional space while preserving distances. Both PCP and dimension reduction techniques have been widely applied in multivariate TF design [1], [2], [4], [9], [33], [46]. However, when PCP is used as the sole design interface, users repeatedly have to adjust parameters for each dimension to define a feature in high dimensional space. It may demand large amount of interactions when the dimensionality is huge. Identifying a feature is easier in MDS plots based on the density of local point clouds. But it is hard to discern physical meaning from the clusters in a MDS plot, as the original numerical distribution information on each dimension is lost after the dimension projection. Based on SPPC (Scattering points into parallel coordinates) [42], which smoothly integrates MDS plots into PCP for multi-dimensional data visualization, we have developed a new multidimensional TF design which leverages and combines the power of PCP and MDS. The proposed design flexibly integrates multiple interactive exploration spaces, including tone-mapped continu-

- *H. Guo, H. Xiao, and X. Yuan are with Key Laboratory of Machine Perception (Ministry of Education), and School of EECS, Peking University, Beijing, P.R. China, 100871.*
  E-mail: {hanqi.guo,xiaohe.pku,xiaoru.yuan}@pku.edu.cn
- *H. Guo and X. Yuan are also with Center for Computational Science and Engineering, Peking University, Beijing, P.R. China, 100871.*
- *To whom correspondence should be addressed. Email: xiaoru.yuan@pku.edu.cn*

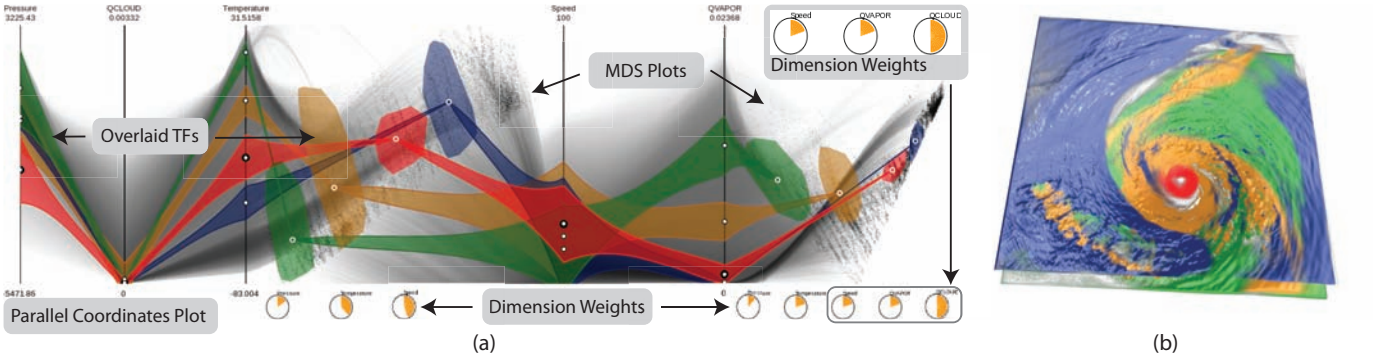


Fig. 1. Volume visualization of a multivariate data set (Hurricane Isabel) with our proposed system. (a) A transfer function is specified in the transfer function design interface of our system, in which multidimensional scaling (MDS) plots are embedded in the parallel coordinates to facilitate feature selection without context switching. (b) Corresponding volume rendered image with the transfer function specified in (a).

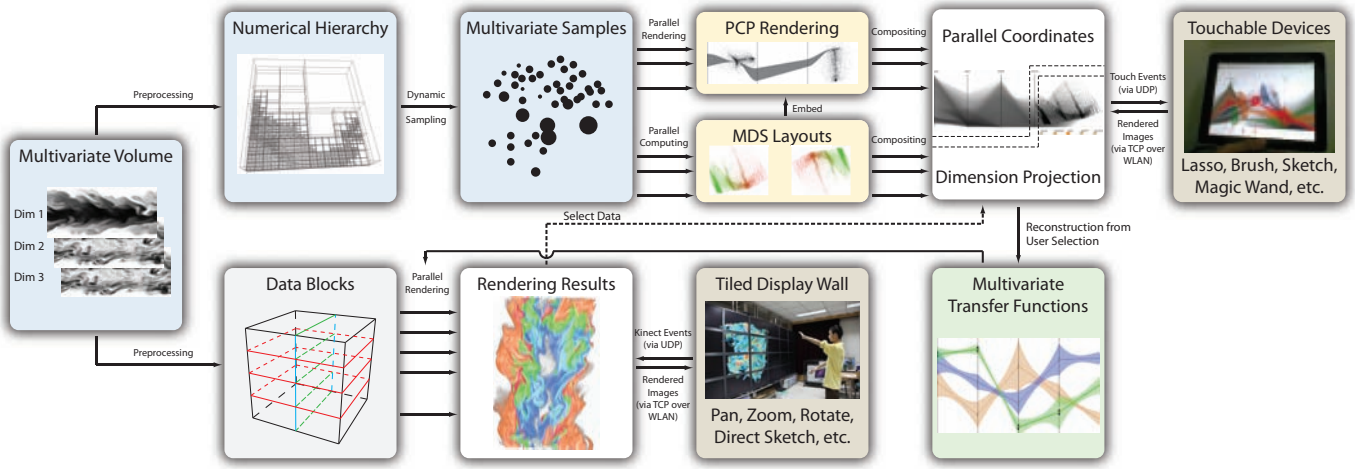


Fig. 2. System pipeline of the proposed multivarite volume data visualization and TF design. A multivariate volumetric data set is first preprocessed with level-of-detail representation before projected to the PCP and MDS plots in the transfer function design interface. Users interact with the plots and classify features of the volume data. TFs then can be constructed based on the user clustering results with user specified colors and opacities. Further interaction can also be performed on the volume rendered images, and fed back to alter the TF design. The volume rendering part together with the drawing of PCP and MDS plots are parallelized in the system.

ous PCP, weight-adjustable MDS plots, and a volume rendered image views with sketch feedback. The seamless integration between PCP and MDS plots provides high efficiency in feature classification by avoiding context switching between different data presentations during the user interaction. In our system, TFs can be defined and modified in different views by various interactions as shown in Figure 1. Different views are all linked to enable synchronization between changes in each individual view.

In this paper, we introduce scalable parallel implementations for PCP, MDS, as well as parallel multivariate volume rendering. The performance of the system is enhanced through leveraging computing parallelism for large data visualization and analysis. Three major components of the system, multivariate volume rendering, adaptive continuous PCPs, and metric adjustable MDS plots are parallelized and integrated into the whole system for acceleration under parallel computing and rendering environments. In our system, users can explore multidimensional feature spaces of large data sets with integrated PCPs and MDS plots, and diagnose interesting features by designing multivariate TFs with the interaction tools provided. In addition, we provide extensions to a tiled display wall and touchable devices for more flexible user interaction and data analysis.

The contribution of this work is two-fold. First, we present an efficient system for visualizing and analyzing multivariate volumetric data through tightly linked views of continuous PCPs, metric adjustable MDS plots, and volume rendering with interactive feedback. Second, in order to facilitate large data visualization, we developed several algorithms including Scalable Pivot MDS (SPMDS), parallel adaptive continuous PCP rendering to enable the system to work in parallel computing environment with high scalability. Overall, we provide a scalable system for multivariate volume data visualization.

The remainder of this paper is organized as follows: we first summarize related prior work in Section 2. Then in Section 3, we give an overview of our proposed system, and we introduce the algorithm details and the extension on parallel visualization in Section 4 and Section 5, respectively. The user interfaces on the tiled display wall and touchable devices are presented

in Section 6. Application cases of the system and the corresponding visualization results are demonstrated in Section 7, followed by the conclusion and discussion of its limitations in Section 8.

2 Related Work

In this section, we review the literature in the field of transfer function design. As PCPs and MDS plots are employed in our transfer function design interface, their related work is also discussed together with parallel volume rendering.

Transfer Function Design is critical for insightful and informative volume visualization, and it is regarded as a key topic in the field. In general, the majority of TF design methods can be categorized into data-centric and image-based [30]. Image-based transfer functions are goal-oriented. For example, TFs and the corresponding rendered images can be gradually optimized and guided by the image quality [13]. Design galleries layout and present the rendered images generated by different TFs for users to select better TF configurations. TF design can also be generalized as a 3D image processing procedure to achieve better volume visualization results [10]. On the other hand, data-centric TF design emphasizes the volume data set itself and the derivative properties. Much work has been focused on designing TFs considering different dimensions for better classification in the visualization results [19]. For example, boundary information can be added to enhance rendering results by accounting for gradient magnitude [21]. The visualization community also proposed various TF design by exploiting curvatures [18], relative sizes [6], and ambient occlusion [7], in order to separate and present more features. Statistical properties can also be extracted as new dimensions for TF design [12], [26].

It is non-trivial to design multi-dimensional TFs without effective user interfaces. Some widgets are designed for 1D, 2D or 3D TFs under the guidance of histograms in feature space [19]. For higher-dimensional feature spaces, many alternative methods other than direct manipulation of feature space have been proposed, including cluster space-based design, machine learning based algorithms, projection space-based methods and PCP-based user interfaces.

Cluster space-based TF design interfaces [36] enable users to work in cluster space, rather than in the TF space. Tzeng and Ma [36] utilized ISODATA clustering to classify the feature space. Furthermore, automatic TF design based on hierarchical clustering was proposed [32]. The cluster space-based method makes it convenient to design multidimensional TFs, but its performance relies heavily on the results of clustering, and limited user participation. In addition to cluster space-based methods, machine learning is considered an important technique for TF design [22]. Artificial Neural Networks (ANN) and Support Vector Machine (SVM) algorithms have also been applied to the TF design for multi-modal volume data sets [35]. Sketches on volume slices act as the sample input of the learning algorithms, and the output of the algorithm is the complicated fitting of a multidimensional feature space. Although machine learning based methods are effective and straightforward, it is difficult to check and verify the physical meaning of the visualization results.

Dimension reduction and projection provide a similarity-based layout for the data. The distance in high dimensional space is embedded into a lower dimensional space. Utilizing dimension reduction and projection for TF design has been investigated [9], [33], [46]. The results of the projection in the form of point clouds are much easier to comprehend and select. The local clustering of the point clouds are likely to represent the features in the original volume data under exploration. Various projection and dimension reduction methods have been exploited to facilitate multidimensional TF design, including ICA [33], PCA [31], SOM [9], LLE [46]. In recent years, the performance of dimension projection has been the focus. Two-phase mapping [28] and piecewise Laplacian-based projection [27] are proposed to improve the performance of dimension projection for massive datasets. Recently, Zhang et al. [45] parallelized the classical MDS for scientific applications. In our work, we employ Pivot MDS [5], which is very fast and stable. Different to previous work on high-performance data mapping, we utilize parallel computing techniques to accelerate the data projection.

Parallel Coordinates have been widely used to visualize multivariate discrete data [16], and some extensions on assisting TF design were proposed in recent years [1], [2], [4], [46]. In those designs, TFs are defined on the dimension axes by assigning numerical ranges [1], [4]. PCPs have also been used for feature selection in particle simulations [17], which is a similar process to TF design. Zhao and Kaufman [46] used PCPs for TF design, and showed the transferred voxel points on the dimension reduction plot. In our design, users can additionally select features on the projection plot, and reflect the TFs on the PCP, providing multiple linked explorative spaces for TF design. The simple integration of PCPs and MDS plots has been applied to colormap design of multi-modal remote sensing images [44]. However, due to the inherent complexity and the overwhelming amount of 3D voxel data, it is difficult to design transfer functions with PCPs.

Parallel Volume Rendering is critical for visualization of large data sets. In most situations, the sort-last method renders the data that are pre-partitioned and assigned to different rendering nodes. The result is a composite of rendered images from each node. Binary-swap [23] has been proposed to balance the overhead of image compositing, with the restriction that the number of processors is restricted to 2^n. More recently, several algorithms were proposed to make compositing more flexible, such as 2-3 swap [41] and Radix-K [29]. In our system, parallelism is advantageous to enable handling data sets with larger sizes.

3 Overview

The road map we design in the multivariate volume visualization system is to provide interactive multiple perspective views for helping users analyzing and comprehend the data. First, PCP is a powerful tool to visualize the data distribution on each dimension, as well as their correlation. We employ adaptive continuous PCP to depict the continuously defined volumetric data. Second, MDS plots can better present the data clusters in low dimensional space, yet the information on

each dimension is lost. By interwinding MDS plots into the PCP [42], the multidimensional data analysis becomes more efficient than with traditional tools. Third, multivariate TFs can be designed with the integrated interface of PCP and MDS to reveal spatial distritution of features in volume rendering results. Users can also get direct feedback while interacting with rendered images by sketching tools. At last, in order to ensure interactivity, scalable implementations are added into the system.

The major components of our system and their logic pipeline are illustrated in Figure 2. First in preprocessing, a level-of-detail representation of the multivariate volume data set is constructed and then projected as continuous PCP with embedded dimension projection plots (Figure 3). The user can define and select clusters based on classification characteristics in the plots. The system reconstructs the TFs automatically based on the user selection and updates the classification results in all linked views accordingly. The new generated TFs are the estimation to the numerical distribution of user selection. After a new TF has been defined, the volumetric data is rendered based on the TF definition. Furthermore, interaction can also be performed on the volume rendered images, to directly remove or add components from the overall TF.

3.1 Transfer Function Widget

A TF layer (in color) and a PCP / MDS layer compose the TF widget (Figure 1 (a)). The TFs are presented with ribbon metaphors in an illustrative manner [25] in the top layer, which is spatially more compact and informative. The system links the central points to extend the illustrative PCP to MDS plots. One component of the TF corresponds to one ribbon. The position and width of the ribbon encode the center and range of TFs, respectively. Users can drag corresponding control points to adjust the ribbon centers, and directly tune the width on each dimension. PCP and MDS plots are linked. All the plots can be updated in real time with the operations on TFs in any individual view.

Between neighboring axes of the PCP, or on the side of the PCP, multiple MDS plots with different dimension weights can be embedded, and visually linked by the curve bonds (Figure 3). Between the MDS plots and the PCP, the clusters can be visually traced through connecting lines, without significant cost on context switching. With this design the users can effectively and efficiently obtain data clustering and data distribution information from PCP and MDS plots.

In the TF widget, the system enables various user interactions. On the PCP, it supports traditional brushing such as axis brushing and angular brushing. Users can select a certain range on one dimension axis with axes brushes. Selecting line segments with similar slopes between two neighboring axes, with angular brushing is suitable for filtering features with high correlations shown on the PCP. In the MDS plots, the lasso tool or the magic wand tool are developed to select features. Those tools can pick up similar neighborhood features near a given point. All user interfaces are coordinated. The selection on both PCP and MDS will be presented on both views.

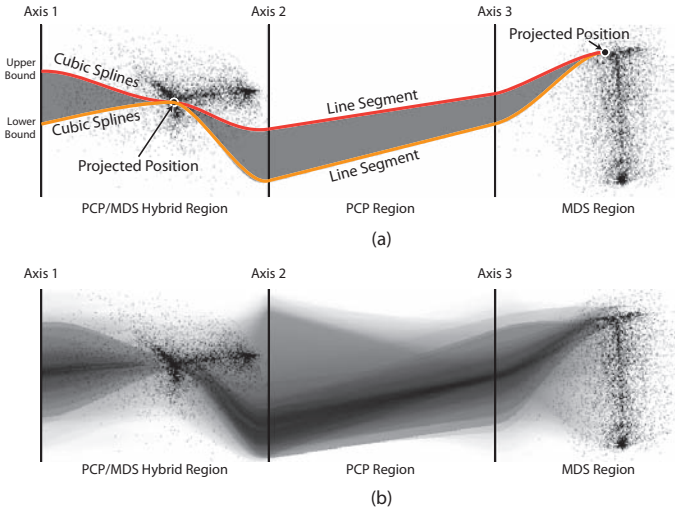


Fig. 3. Smooth linkages between the continuous PCP and the MDS plots: (a) the local shape of the linkage when a subset of samples is shown; (b) the linkage of the whole domain.

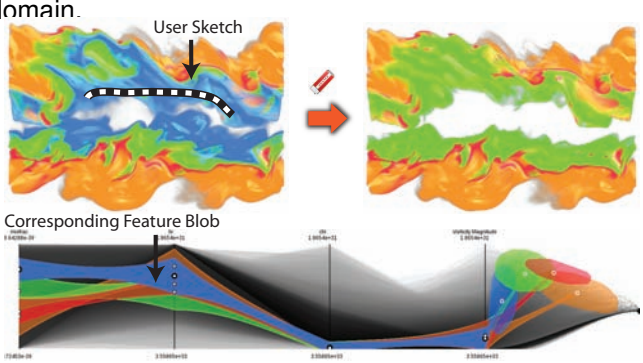


Fig. 4. User interactions on direct volume rendering view. The blue colored TF component is highlighted after a sketch on the rendering view. It can be further removed if the eraser tool is selected.

3.2 Volume Rendering View

Direct sketching interactions are implemented on the volume rendering view in addition to the operations of the TF widget (Figure 4). Users can pick the query tool, and then sketch on desired regions of the rendered volume image. Then the corresponding TF component is selected and highlighted on the TF widget. Users can drop a TF component quickly with the eraser tool. When reviewing the rendered results, the sketch interaction in our system can give instant feedback.

4 Algorithm Details

To support interactive multivariate data visualization and analysis with PCP, MDS and direct volume rendering in our system, we developed corresponding algorithms for each visual component. First, we introduce how adaptive continuous PCP and MDS are generated and rendered, as well as how MDS plots are embedded into a PCP. Secondly, a multivariate transfer function construction algorithm is proposed in order to visualize the selected features on the volume rendering

view. Third, we present the feedback mechanism between the coupled views.

4.1 Adaptive Continuous Parallel Coordinates

PCP, an effective way to visualize discrete multivariate data, maps the high-dimensional points into polylines intersecting parallel axis. However, for continuous data, specifically volumetric data, such mapping may not properly present data features if inappropriate sampling strategies are applied. To solve this problem, a dense presentation of PCP, which is so-called continuous PCP [14] is empolyed to reveal insights into the multivariate volumetric data. Although continuous PCP can suitably present the continuously defined data, a mechanism is lacking to map the features on the continuous PCP back to the original data points. This problem prevents linked brush operations on continuous PCPs.

To address this issue, we developed an adaptive hierarchy rendering strategy for continuous PCP based on BSP (Binary Space Partitioning) tree, which preserves sufficient discrete information for the linked brushes. Other linked components, such as MDS plots, also utilize such discrete information for data projection. The adaptive rendering of the PCP is analogous to the adaptive hierarchical continuous scatterplot [3]. During rendering, the PCP is obtained by accumulating the height map from eligible BSP tree leaves as quadrilateral strip primitives, giving the tolerance error for rendering.

4.1.1 Adaptive Rendering of Continuous PCP

Since there are many homogeneous regions in multivariate volume data, the data scale can be largely reduced by merging isotropic regions with BSP tree structure, in order to support interactive exploration. In each leaf node, the numerical properties can be considered identical. The level of data abstraction with BSP tree can be controlled according to different demands of data analysis. When higher level distributions are under focus, faster rendering results with less details are provided. Moreover, fine features are presented when exploring in details. Such hierarchical sampling strategies provide options and balance for faster visualization and more details with controllable error levels.

The rendering of adaptive continuous PCP is a two-step procedure. First, the leaves of BSP tree are filtered out by tree traversal according to a user defined error threshold. Secondly, the selected tree leaves are accumulated on the plane of parallel coordinates as quadrilateral strips, before being accumulated onto the image buffer. Notice that the intensities of the quadrilaterals are decided by the volume of the corresponding leaves. The splatting and accumulation processes are performed on the GPU (Graphics Processing Unit), in order to achieve interactivity. The rendering error of the adaptive method is controlled by given thresholds (Figure 5). Based on the experimental results, we can roughly expect that the results of the adaptive continuous PCP approaches more accurate values when the dataset is partitioned small enough.

4.1.2 Tone-Mapping of the Adaptive Continuous PCP

Since the accumulation result of the continuous PCP is a float-precision density map with high dynamic range, proper

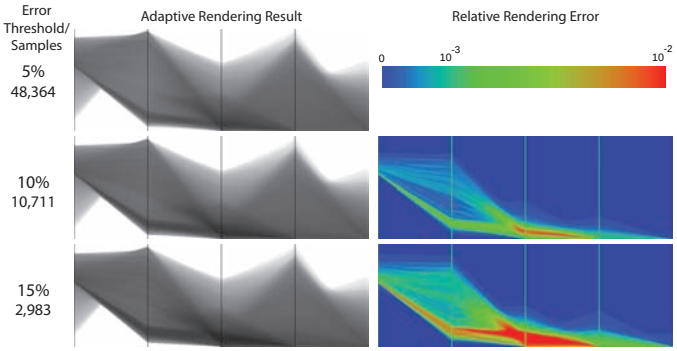


Fig. 5. Comparisons of the adaptive PCP rendering at different user defined error levels. The relative rendering error is scaled with a pseudo colormap.

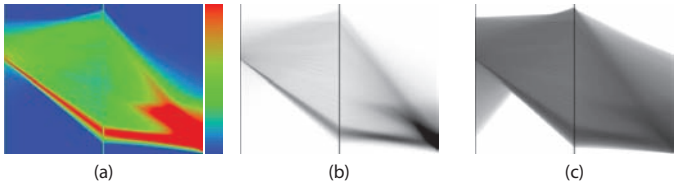


Fig. 6. Result comparisons with linear mapping and logarithmic mapping: (a) pseudo color and the colormap on $[0,1]$; (b) linear; (c) logarithmic.

methods should be applied to prevent information loss of mapping the density map into limited gray scales for displaying. If a naive linear mapping is employed, many subtle yet important features will be lost in the final image. Such situations often occur in high dynamic range data visualization applications [43]. In order to maintain and enhance the feature presentation on the continuous PCP, we utilize a non-linear tone-mapping operator (TMO) for the mapping:

$$I' = 1 - \frac{ln((1-e^{-\alpha})I + e^{-\alpha})}{-\alpha}, \quad (1)$$

Where I' is the intensity of tone-mapped pixel, and I is the original intensity value. The parameter α can be interactively changed by users with the mouse wheel, to achieve different visual enhancements. Compared to linear mapping (Figure 6), the proposed mapping strategy provides superior visualization on small, yet important features. In Figure 6(c), the features on the lower half of the leftmost dimension are clearly visible, but the corresponding feature is totally submerged when linear mapping has been applied, as shown in Figure 6(b). The tone-mapping post-processing is also performed on GPU for interactive exploration.

4.2 Dimension Projection Plots

MDS techniques project points in high dimensional space $\mathbb{R}^k$ to point clouds in lower dimensions (usually 2D) to enable easier feature identification and classification by users. Such embedding process is based on the dissimilarity values between every two points, which is usually defined by proper distance metrics in high dimensional space. The distances are

approximately preserved during the dimension reduction by minimizing the errors, such that $\delta_{ij} \approx \|x_i - x_j\|$:

$$\min \sum_{i<j} (\|x_i - x_j\| - \delta_{ij})^2, \tag{2}$$

where $\delta_{i,j}$ is the high dimensional distance between point i and j, and $\|x_i - x_j\|$ is the distance in embedded dimensional space. The layout of the MDS plots changes with different distance metrics, and each individual plot conveys different perspectives of the multivariate data.

In our system, users can define and adjust various distance metrics on multiple MDS plots. Two facets of distance metrics include norm type and weights of dimensions. Commonly used norms, e.g. Euclidean norm (2-norm), Manhattan norm (a.k.a. city blocks, 1-norm), and infinity norm (inf-norm) can be adopted to generate MDS layouts with different insight. On the other hand, the impact of each individual dimension can be boosted or reduced by tuning its weight. Domain knowledge can be utilized to generate meaningful results. Meanwhile, animated MDS layouts during the tuning process help to understand the structure of the complicated high dimensional space.

4.2.1 Pivot MDS

We utilize a low storage and low computational method called Pivot MDS [5] for dimension projection, which is a sampling-based progressive approximation of classical MDS [34]. In classical MDS, inner products rather than distances in Eq. 2 are used. Denote the points in the embedding d-dimensional space as a matrix $\mathbf{X} \in \mathbb{R}^{m \times d}$ with $\mathbf{X} = (x_1, \ldots x_m)^T$, and $\mathbf{B} = \mathbf{X}\mathbf{X}^T$, Eq. 2 can be transformed to a matrix form:

$$\min \sum_{i,j} (\Delta^{(2)} - \mathbf{B}), \tag{3}$$

where $[\Delta^{(2)}]_{ij} = \delta_{ij}^2$, and $\mathbf{B}$ is the double-centered matrix of the squared dissimilarity matrix Δ. The minimum in Eq. 3 can be achieved when $\mathbf{X} = \mathbf{V}_d \Lambda_d^{\frac{1}{2}}$, where $\mathbf{V}_d \in \mathbb{R}^{m \times d}$ and $\Lambda_d \in \mathbb{R}^{d \times d}$ are the d largest eigenvectors and eigenvalues w.r.t $\mathbf{B}$ [5]. In Pivot MDS, the double-centered matrix $\mathbf{B}$ is replaced by a double-centered $m \times k$ smaller submatrix $\mathbf{C}$ of squared distances Δ, assuming $\mathbf{B}^2 \approx \mathbf{C}\mathbf{C}^T$ [5]. Thus, the computational complexity of the double-centering matrix is decreased from $O(n^2)$ to $O(nk)$, where k is the number of pivot elements. Meanwhile, the problem size of the eigensolver is reduced to $k \times k$, which is far less than that of classical MDS (typically $k = 50, 100$). The serial version of Pivot MDS can be summarized as follows:

- Randomly pick k pivot items from the input data set;
- Calculate the squared distance between each point and all pivot points, and store them in matrix Δ;
- Construct the double-centered dissimilarity matrix $\mathbf{C}(c_{ij})$ between pivot items and all input items, whose elements are defined as

$$c_{ij} = -0.5(\delta_{ij}^2 - \frac{1}{m}\sum_{r=1}^{n} \delta_{rj}^2 - \frac{1}{k}\sum_{s=1}^{k} \delta_{is}^2 + \frac{1}{mk}\sum_{r=1}^{n}\sum_{s=1}^{k} \delta_{rs}^2), \tag{4}$$

- Calculate the eigenvalues and eigenvectors of the matrix $\mathbf{C}^T\mathbf{C}$

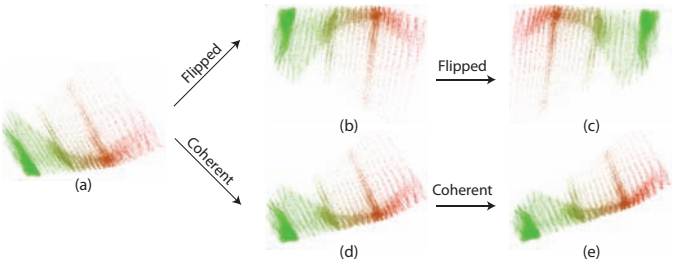


Fig. 7. Rotation correction for the MDS results. The weighting values modify the distance metric of the MDS, thus adjusting the MDS layout. The weighting of dimension "Temperature" is set to 30% in (a), and then it is increased to 35% in (b) and (d), 40% in (c) and (e). The MDS layouts are flipped randomly in traditional methods as shown in (b) and (c). Sequence (a), (d) and (e) keep coherence with our rotation correction algorithm.

- Pick up the largest d eigenvectors $\{\mathbf{v}\}$. The low dimension embedding is achieved by

$$\mathbf{x}_i = C\mathbf{v}_i,\ i \in \{0,\ 1,\ 2,\ \ldots,\ d\} \tag{5}$$

where k is the number of pivot points, and n is the total number of data points. δ_{ij} stands for the dissimilarity of the ith item of input data with the jth pivot item.

4.2.2 Rotation Correction for MDS Plots

Due to the numerical instability and random factors in Pivot MDS, a rotation correction algorithm is applied to ensure frame coherence during tuning the dimension weights. There are two steps in the algorithm. First, the same set of pivots are used whenever the MDS plot updates, in order to avoid random factors related to the first step of the Pivot MDS algorithm. Secondly, a rigid transformation is applied to every new result, to minimize the rotations between the new result and the former one. We followed the least-squares registration algorithm [37]. Specifically, our method searches for the zoom factor c, rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}$ that satisfy the following condition:

$$\arg\min E = \sum_{i=0}^{n} ||\mathbf{p}_i' - (c\mathbf{R}\mathbf{p}_i + \mathbf{t})||^2, \tag{6}$$

where $\{\mathbf{p}_i'\}$ and $\{\mathbf{p}_i\}$ are the newly generated and the original point cloud. Notice that we must use a rigid transformation instead of non-rigid linear transformation, because non-rigid transformation cannot keep the relative distances of the projected points. After finding the parameters by optimization, the transformed points $\{\mathbf{p}_i''\}$ are:

$$\mathbf{p}_i'' = c^{-1}\mathbf{R}^{-1}(\mathbf{p}_i' - \mathbf{t}). \tag{7}$$

Figure 7 shows an example in which an MDS plot in (b) and (c) flipped randomly when the weights of one dimension varied from 30% to 35% and 40%, respectively. When applying our rotation correction algorithm, the MDS plots are stable as shown in Figure 7(d) and (e).

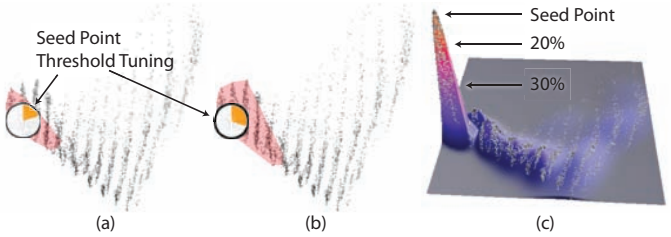


Fig. 8. Magic wand tool for user selection on projection plot. A threshold can be defined by users for automatic diffusion based on the point distribution shown in (c).

4.3 Integrating MDS and PCP

To maintain visual continuity and avoid context jumps, SPPC [42] is adopted in our system. In the work of SPPC, regions between two selected neighboring coordinate axes are converted into a scatterplot or MDS directly. Smooth transition between the newly created MDS plot with the rest part of parallel coordinates is achieved by crossing each MDS plot point with a curve connecting the corresponding ploy line(s) in the parallel coordinates. We substituted the spline curves from original SPPC with quadrilateral strips. In this way, the accumulated intensity of the spline strips remains coherent to the strips from the PCP, and the data can be tracked continuously from axis to axis. On PCP axes, the width of the spline strip of each BSP tree leaf is set to be the range on the specific dimension, as described in Section 4.1.1. In the MDS plots, the strip length passing through the corresponding point is set to zero. Cubic splines connect the upper and lower boundaries on the axis to the projected point in the MDS plots as shown in Figure 3. Notice that the MDS plots can also be embedded on the side of the PCP, which is better for occlusion reduction, but not intuitive to present correlations. Users can freely decide and combine the components.

The embedding of MDS plots into the PCP brings multiple benefits. First, the combination provides strong visual hints of data clusters [15], [42]. Furthermore, the tool supports coordinated selection between the PCP and the MDS plots without the cost of context switching.

4.4 Feature Selection on the Linked Views

Our proposed framework provides several linked spaces for exploration, including the PCP, the MDS plots, and the volume rendering view. Each component provides a distinctive capability for feature selection. The interactions on the PCP are similar to those in the previous work, e.g. axis brushing and angular brushing, so we mainly introduce here the details for the interaction with the MDS plot and the volume rendering view.

4.4.1 Interactions on MDS Plots

Two interaction methods, the lasso and the magic wand tool, are available on the MDS plots. The lasso tool picks up the sample points inside a specified polygon. The magic wand tool automatically selects the neighborhood samples from a given seed point according to the local distribution in the 2D projection space as illustrated in Figure 8. A fuzzy weight (similarity value) is given to each point, in order to estimate the numerical distribution for the TF construction.

Computing the magic wand selection has two phases, including the height map construction and the diffusion process. The height map is essentially the construction of the feature space [24] in the 2D projection space, which can be generated by GMM fitting or point splatting. An example result for height map construction is shown in Figure 8(c). We discretize the 2D space as an image. For the diffusion process, we first find the nearest data point from the user defined position as the diffusion seed, and then utilize a BFS (breadth-first search) process that spreads the selection from the seed point to the final selection set by searching the neighborhoods. At every single step, the algorithm gives a fuzzy weighting depending on the similarity value with the seed point.

4.4.2 Sketch Interaction on Volume Rendering View

One key issue of direct operations on the rendering view is the intelligent speculation of user intention. In our design, users can direct pick up what they see from the results, which aims to find out the closest Gaussian blob from the TF set. A ray can be traced from a certain sketch point, and we can calculate the visibility of each sample point along the ray. The visibility at position $\mathbf{x}$ can be written as [8]:

$$V(\mathbf{x}) = (1 - \alpha(\mathbf{s}(\mathbf{x})))O(\mathbf{x}), \tag{8}$$

where $\mathbf{s}(\mathbf{x})$ is the multidimensional sample at point $\mathbf{x}$, and α is the opacity TF. $O(\mathbf{x})$ is the occlusion value at $\mathbf{x}$. We assume that the user intention is very likely to fall on the local maxima of the visibility. We can also derive the ambiguity of the visibility local maxima A_i. Finally, the Gaussian blob selection i is decided by searching for the optimized value of $A_i V_i$. In order to accelerate the ray traversal, the BSP tree traversal along the ray has been implemented.

4.5 Transfer Function Construction

The construction process converts user selected features with assigned colors and opacities into TFs. The major algorithm for the construction in our system is the Gaussian Mixture Model (GMM), which estimates the input features. A series of Ellipsoid Gaussian TFs [20], [38] are generated as the construction results. Notice that the extracted Gaussian blobs in the GMM are only the estimation to the user selection, instead of the original values.

Suppose the numerical distribution in every BSP tree leaf is Gaussian. Ideally, the Gaussian blobs from selected leaves can be taken as TFs. However, due to limited computational capability, the sum of huge numbers of Gaussian function values for each sample point cannot be evaluated in real-time. More efficient fittings for the user defined features with a relatively small numbers of mixtures was developed in our system, since many of the leaves selected will have similar numerical distributions. We employed the GMM to fit the feature space:

$$\sum_{k=1}^{m} \mu_k G_k(\mathbf{v}) = \sum_{i=1}^{n} \omega_i m_i G_i(\mathbf{v}) + \varepsilon(\mathbf{v}), \tag{9}$$

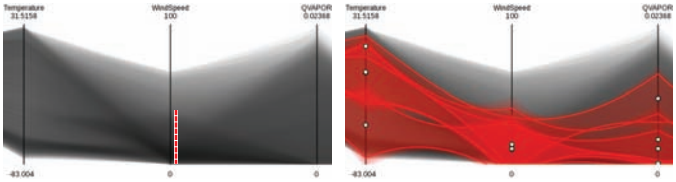


Fig. 9. The transfer function generation: (a) a user defined stroke on the PCP; (b) the constructed transfer function with 3 Gaussian blobs.

where G_i, m_i, ω_i are the estimated Gaussian distribution of BSP tree leaf i, the volume of the leaf, and the fuzzy selection coefficient, with the range of $[0,1]$. The left term of the equation is the Gaussian mixture, where m is the number of Gaussian blobs, and μ_k is the weight for the Gaussian blob G_k. The standard Expectation Maximization (EM) algorithm determines the parameters of the GMM model through iterations. The Gaussian mixtures can be transformed to Gaussian TFs [20] by multiplying the opacity value α_{max}:

$$GTF(\mathbf{v}) = \alpha_{max} \sum_{k=0}^{m} \mu_k G_k(\mathbf{v}), \tag{10}$$

where $\mathbf{v}$ is the input data vector, and μ_k is the maximum opacity value for scaling each Gaussian bolb, which is obtained in Eq. 9. The final color $\mathbf{C}$ and the opacity α values are:

$$\mathbf{C} = \frac{\sum \alpha_i \mathbf{C}_i}{\sum \alpha_i}, \alpha = \sum \alpha_i, \alpha_i = \alpha_{max}\mu_i \tag{11}$$

where $\mathbf{C}_i$ and α_i are the color and opacity of each Gaussian component. Gaussian TFs are applied on-the-fly to each sample point in the standard GPU volume raycasting process.

5 Parallel Visualization System and Performance

In order to support larger dataset exploration with our method, we accelerate the user interface by leveraging parallel computing techniques. There are three major user interaction components in our system, including the computing of MDS layouts, the rendering of a PCP, as well as the multivariate volume rendering. We have added three corresponding parallel subsystems to the pipeline (Figure 2) to make it possible to visualize and analyze larger datasets with our system. In a parallel environment, data is distributed to different types of processes, and then results are gathered back to the GUI processes before the GUI sends out requests during user interaction.

5.1 Parallel MDS Computing Subsystem

A scalable MDS layout solution is provided for large dataset exploration in our system, by exploiting parallel computing technologies. Although there are many existing MDS techniques, the performance and the scalability of the algorithms are limited when dealing with large dataset. For example, the computational complexity of classical MDS (CMDS) and many iterative force-directed distance scaling methods are $O(n^2)$, which prevents interactivity if the data set is large. To project high dimensional points into 2D effectively, we propose Scalable Pivot MDS (SPMDS) to scale for large data.

There are four major sub-procedures that can be parallelized in Pivot MDS, including the computation of the squared distance matrix $\Delta^{(2)}$, the double centered sub-matrix C and its inner product $\mathbf{C}^T\mathbf{C}$, as well as the eigensolver. As illustrated in Figure 10, suppose there are N parallel processes, each process p $(p = 1,2,\ldots,N)$ holds a part of the multivariate data $\mathbf{V}_p \in \mathbb{R}^{m_p \times n}$ and all the pivot elements $x'_0,\ldots,x'_{k-1}$. The algorithm for each process can be written as follows:

- Calculate the sub- squared distance matrix $\Delta_p \in \mathbb{R}^{m_p \times k}$. Each element $\delta_{p,i'j}$in Δ_p is the squared distance between the point i' and the pivot point j;
- Calculate the column summation $\mathbf{S}_{p,i'} = \sum_{s=1}^{k} \delta^2_{i's}$, and the row summation $\mathbf{S}'_{p,j} = \sum_{r=1}^{m_p} \delta rj^2$ of matrix Δ_p. The column summation of the whole matrix Δ is $\mathbf{S}_i = \mathbf{S}_{p,i'}$, after the transformation of index i'. The row summation Δ is the reduction result of each process: $\mathbf{S}'_j = \sum_p \mathbf{S}'_{p,j}$. The total summation $\mathbf{S}''$ of Δ can be further obtained by summing values in $\mathbf{S}'_j$;
- Construct the sub-double centered dissimilarity matrix $\mathbf{C}_p$, whose elements are defined as

$$c_{i'j} = -\frac{1}{2}(\delta^2_{p,i'j} - \frac{1}{m}\mathbf{S}_{p,i'} - \frac{1}{k}\mathbf{S}'_{p,j} + \frac{1}{mk}\mathbf{S}''). \tag{12}$$

- Calculate $\mathbf{C}_p^T\mathbf{C}_p$, and reduce the sum of $\mathbf{C}_p^T\mathbf{C}_p$ in all process into the first process as $\mathbf{C}^T\mathbf{C}$. (It is straightforward to prove $(\mathbf{C}_1,\ldots,\mathbf{C}_N)^T(\mathbf{C}_1,\ldots,\mathbf{C}_N) = \sum_{p=1}^{N} C_p$);
- Solve the largest d eigenvalues and eigenvectors by utilizing standard scalable linear algebra libraries (e.g. ScalLAPACK), and distribute the eigenvector $\mathbf{v}_d$ to each process.
- Obtaine the projected d-dimensional coordinates by multiplying the sub-double centered matrix and the eigenvectors:

$$\mathbf{X}_p = \mathbf{C}_p\mathbf{v}_d. \tag{13}$$

5.2 Parallel PCP Rendering Subsystem

The performance of the adaptive continuous PCP rendering can be boosted by leveraging the power of parallelism. The data is distributed and scattered to different processes, and each node generates a partial density map for the PCP. The final PCP is obtained by gathering and reducing the results from the rendering nodes. Currently the cross-node summing and reducing process is performed by employing reducing functions in MPI (Message Passing Interface), and the performance can be improved by utilizing GPU acceleration on each node.

There are two strategies for partitioning the data for parallel processes. One method is to assign a spatial region of the original data to each node, and the other method is to assign equivalent BSP tree leaves to each node. In practice, the load balancing of the latter method is better.

5.3 Parallel Volume Rendering Subsystem

Sort-last strategies, which have been widely applied in parallel volume rendering, are utilized in our system. The whole multivariate volume data set is subdivided into subregions of equal

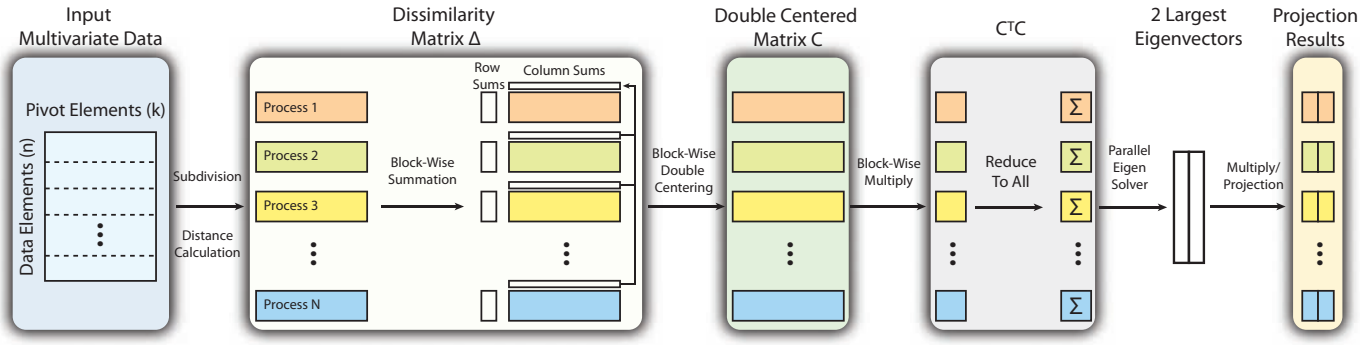


Fig. 10. The pipeline of SPMDS (Scalable Pivot MDS). The input multivariate data points are subdivided and distributed into parallel processes, before the dissimilarities are calculated in sub-matrices. After the row sums and column sums are derived, the double-centering sub-matrices C_i are calculated in parallel. The summation of the dot products C^TC is reduced from the corresponding results $C_i^TC_i$ in each process. The final projection results are obtained after multiplying each dissimilarity sub-matrix and the two largest eigenvectors of C^TC.

size before they are assigned to specific rendering processes. If the number of processes is a power-of-two, the whole volume is divided into octree leaves. Or it is divided as equal slabs along a specific axis. The final images are obtained by compositing the rendered images from all processes in proper orders. Currently, Radix-K image compositing algorithm [29] is utilized in the system.

We have implemented both a GPU-accelerated and a pure-software parallel rendering version. In the GPU implementation, the block rendering and the cross-node image compositing are both accelerated by graphics cards. During the block rendering, the parameters of the multivariate Gaussian transfer functions are stored as textures. The values of Gaussian functions can be looked up from a pre-computed 1D texture, which is about 4 times faster than the on-the-fly call of the exponential functions in a shader program. The communication cost is higher than pure software rendering due to the extra data swapping from the video memory to the main memory. On the other hand, the pure software implementation is based on hybrid MPI and OpenMP parallelism. On each node, the image space is divided into parts for OpenMP threads to perform block rendering and image compositing.

On each rendering process, standard front-to-back raycasting technology is utilized for rendering. If the data size of the assigned region is too large to handle, out-of-core strategy is leveraged to break through the limitation. In out-of-core mode, the subregion is further subdivided into blocks in equal storage sizes, which can fit into the video memory. At first, the rendering process allocates memory for one block, and then dynamicly loads, renders, and dumps each block one after another. Due to the limited bandwith between the main memory and the GPU, the performance drops because of the swapping of the block data. In order to accelerate the framerate during interaction, low resolution blocks are also generated for faster rendering.

5.4 Performance

The system is mainly in C++, and part of the numerical subroutines are implemented with Fortran and NVidia CUDA. The graphical components are implemented with OpenGL and Cg shading language. The multivariate volume rendering is implemented in both the hardware accelerated version and the pure CPU version, which is prepared for clusters without graphics hardware or in-situ purposes. MPICH2 is utilized for cross node communication. In the parallel environment, different types of processes are dynamically spawned according to the host configuration at the startup of the system, and then they are partitioned into different communication groups.

The performance benchmark (Figure 11) is tested both on a single workstation and in parallel environments. The former benchmark is performed on a Dell T3400 workstation, with a 2.66GHz Intel E7300 CPU, 4GB memory, and an NVidia GTX 470 graphics card with 1280MB video memory. The parallel platform is an 8 node SuperMicro PC cluster with a Dell T3400 workstation as the dedicated GUI node. Each cluster node has two Intel Xeon E5520 CPUs in 2.26MHz, 48GB main memory, and two NVidia GTX 275 graphics cards with 768MB video memory. The nodes are connected with Gigabit Ethernet. The GUI node has an Intel E7400 CPU in 2.80GHz, 2GB memory, and an NVidia 275 graphics card. The operating system of the cluster is 64-bit CentOS 5.6 Linux. Currently, we only utilize affordable and frequently used Ethernet for communication. The performance and the scalability will be increased if a faster network environment is available.

5.4.1 *Performance on the Single Workstation*

On the single workstation, three different multivariate data sets are tested, including hurricane Isabel simulation, the ionization front instability simulation, and the turbulence data. The data properties and the timings of each subroutine are shown in Table 1. In Table 1, T_{pre}, T_{proj}, T_c, T_r are timings of preprocessing, projection, TF construction and frame update time, respectively. For interactive exploration, the most important timings are T_{proj} and T_r. The low T_c timing can insure instant feedback after users' query and a selection on both PCPs and MDS plots. Although a T_{pre} is relatively long, users can instantly obtain feedback during a data exploration since preprocessing is offline.

Currently, the preprocessing is available only on the single workstation. Out-of-core data reading strategy is applied when creating the hierarchical numerical data structures, if the data

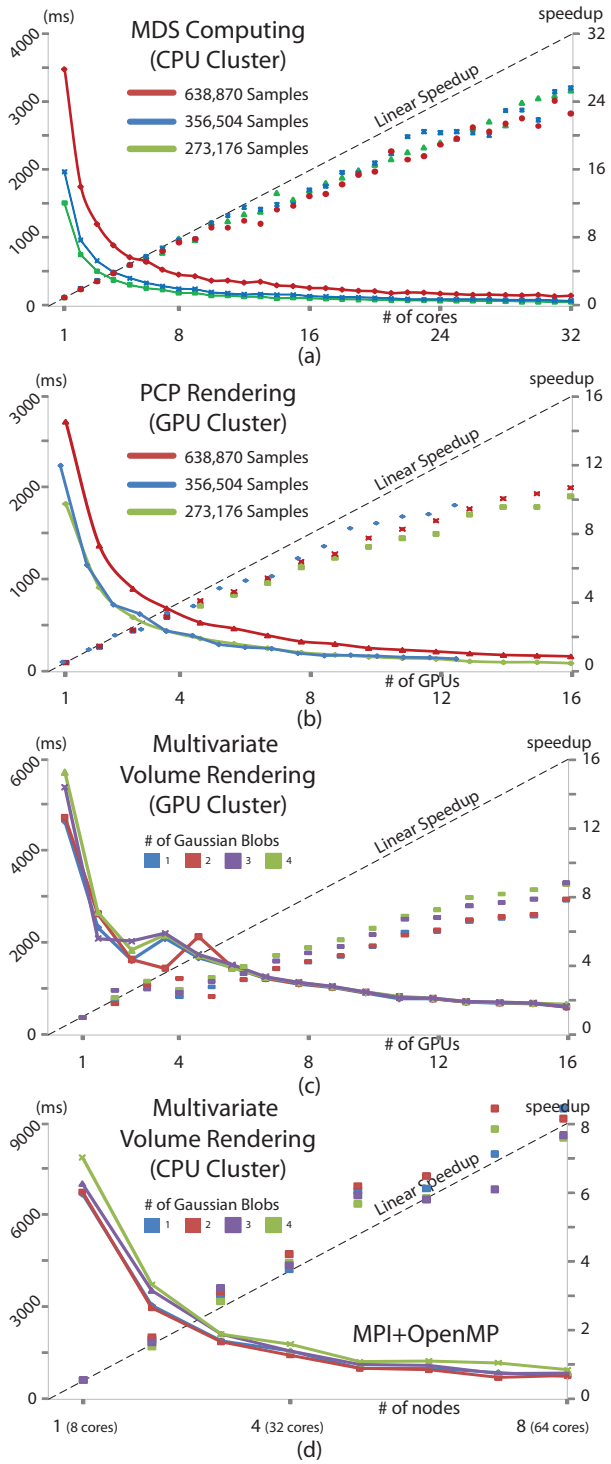


Fig. 11. Benchmark of parallel MDS computing (a), parallel PCP rendering (b), parallel multivariate volume rendering with GPU cluster (c) and CPU cluster (d) on the turbulence data, using different parameters and cores.

is larger than the main memory. The out-of-core scheduling and trashing mechanism is based on data blocking, which is similar to the out-of-core rendering (Section 5.3). The processor loads one BSP tree leaf once to build its numerical hierarchy, and then dumps it before loading another leaf. In general, if the data is larger and more complicated, more leaves are generated, which determines the performance of T_{proj} (also refer to Figure 11). T_c is not only related to the number of leaves, but also decided by the features that the user selected. If the feature is more complicated, the iteration steps of GMM will take longer (c.f. Section 4.2).

Data Set	Size	Dim	T_{pre} (s)	leaves	T_{proj} (ms)	T_c (ms)	T_r (ms)
Isabel	$500^2 \times 100$	5	224.0	55,273	150	725.0	233
Ionization	$124^2 \times 300$	12	90.6	40,321	148	1045.5	833
Turbulence	$1,000^3$	3	$\sim 4hrs$	638,870	1,688	2618.0	N/A

TABLE 1
Timings on single workstation.

5.4.2 *Performance on Parallel Clusters*

The parallel MDS subsystem mainly relies on the computational capability of CPUs. Numerical libraries, including BLAS and LAPACK are utilized for matrix and vector computation. We tested the performance of SPMDS on the turbulence dataset with various numbers of cores (Figure 11(a)). Different sample numbers are filtered out from the hierarchy data structures of the data with different error thresholds. There are more sample points if the corresponding threshold is lower. The pivot number k of SPMDS is set to be 50. Besides the initial data sending and final result gathering time, SPMDS is highly parallized. Several factors may negatively impact the performance. Firstly, the result gathering routine, which is essential for visualization, is not very scalable. The network response time and the transmission rates also influence the performance.

The performance of the parallel PCP subsystem is shown in Figure 11(b). The testing is performed on different numbers of GPUs on the cluster, ranging from 1 to 16. The total time of the PCP rendering decreases as more processes are used. If the number of samples is larger, the speedup is higher. When the data sizes are small, low speedup is observed due to the communication cost and the cost of reduction subroutines.

The timings for the rendering subsystem in both hardware-accelerated version and pure-software implementation are shown in Figure 11(c) and (d). The raycasting step-size in the test is 1 voxel. All the rendered image sizes are 640x480. Different numbers of Gaussian blobs (from 1 to 4) in the multivariate TF are tested. From the results, we can observe that the rendering time is longer when more TF components are utilized. The performance of the rendering increases as more computational resources are utilized. Since the storage size of the turbulence data is about 12GB, the out-of-core mode is employed for rendering, if the block on each process is larger than the video memory. Frequent swapping on the video memory and main memory reduces the performance of rendering. On the contrary, in-core mode saves time if the storage size on each process is less than the available memory.

6 User Interaction on Tiled Display Wall and Multitouch Devices

The proposed system also provides flexible user interaction with both tiled display wall and touchable devices. Tiled Display Walls (TDW) can present the rendering result in

very high resolution, which provides more details of the data. Touchable devices, such as an iPad, can provide a flexible user interface for data analysis. In our system, we combine both powerful visualization and interaction devices to facilitate data analysis. As shown in Fig 2, the TDW displays the multivariate volume rendering result, and the transfer function design interface with PCP and MDS is provided to users on a iPad. On the TDW, users can pan, zoom, and rotate the 3D view by motion and gesture, which are captured by Microsoft Kinect device. Meanwhile, the transfer function of the volume rendering result can be modified with the iPad interactively. With such configuration, users are free to change views of the volume rendering on the TDW, while flexibly exploring the parameter space with the iPad.

Both TDW and iPad are driven by the visualization cluster. The volume rendered images are composited and transfered to each tile with MPI, and the rendered PCP with MDS images are transmitted to the iPad via TCP packets over wireless LAN. On the other hand, the user interaction events are carried to the cluster with UDP packets. Multitouch events are directly sent by the iPad, and the 3D positions of the skeleton are extracted based on the color and depth images that are captured from Kinect cameras.

7 Case Analysis

In this section, we demonstrate the capability of our system with multivariate TF design through applications on several representative data sets listed in Table 1.

7.1 Atmospheric data set (Hurricane Isabel)

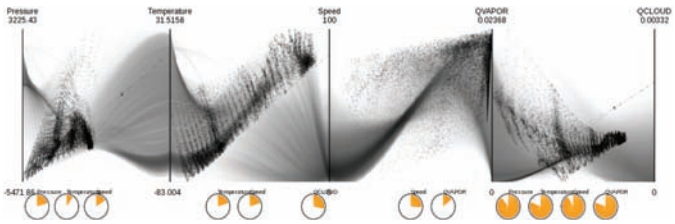


Fig. 12. Different MDS plots of Isabel data embedded in the PCP. The impact of each dimension is assigned by the pie-chart-like round widget at the bottom of each MDS plot.

The Hurricane Isabel dataset is an atmospheric simulation (IEEE Visualization contest 2004, http://vis.computer.org/vis2004contest). In this case, five dimensions are considered, including pressure, temperature, wind speed magnitude, water vapor mixing ratio (QVAPOR), and cloud moisture mixing ratio (QCLOUD). From the different perspectives and components provided by the system, users can better comprehend the data. In MDS plots, data clusters are easy to be identified (Figure 12). By tuning the weight of each dimension on the small round widgets below the PCP, the MDS plots give animated results, which helps users to comprehend the role of each dimension. Sub-dimensional space can also be navigated by selecting a small set of dimensions. The PCP presents the numerical distributions on each dimension, as well as the correlationship between neighboring dimensions.

The exploration of the data is very helpful for users to understand the multivariate features in the data. Figure 14 shows the exploration process of the hurricane eye. Color and opacity values can be assigned to desired clusters by brushing, lasso or magic wand tools on both PCP and MDS plots. After further tuning the Gaussian blobs in the PCP view, insightful direct volume rendering results are generated. In Figure 14, we first assume that the wind speed of the hurricane region is relatively high, and then we brush on the corresponding range on the PCP axis. The volume rendering result which reveals the shape of hurricane eye is shown after the selection. Further hypothesis and exploration can be done by tuning the numerical ranges on the PCP.

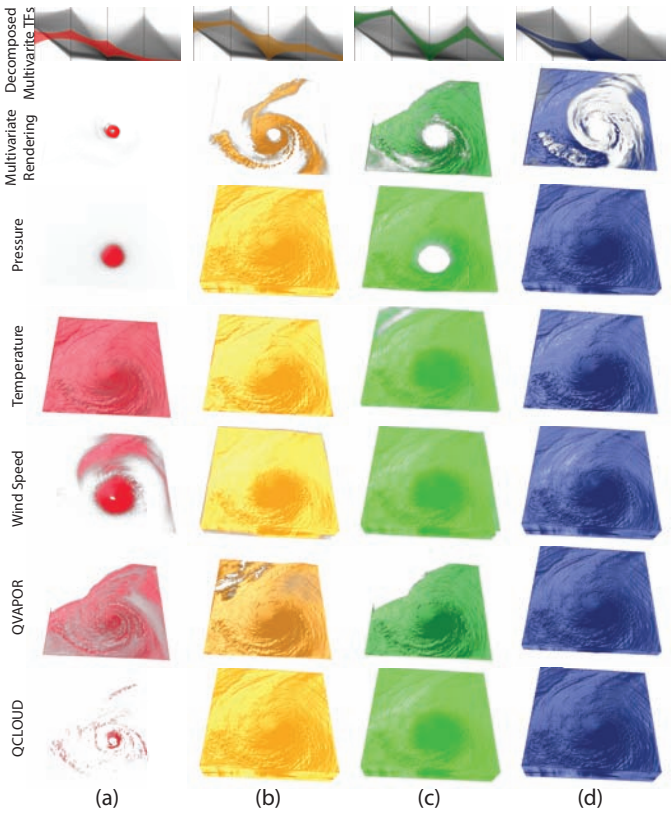


Fig. 13. Multivariate and univariate rendering results of Hurricane Isabel with 4 decomposed multivariate transfer functions.

After several trial-and-error steps, insightful rendering results can be obtained (Figure 1). Rendering results of each individual dimension, as well as the multivariate rendering results of the decomposed TF are shown in Figure 13. The red part of the result is the side region of the hurricane eye, where the pressure is low, and the values of wind speed and QCLOUD are medium; The outside feature with yellow color has a much higher pressure value but lower wind speed than the eye region. We can also see how different features are mixed. Many other patterns can be recognized from the visualization results.

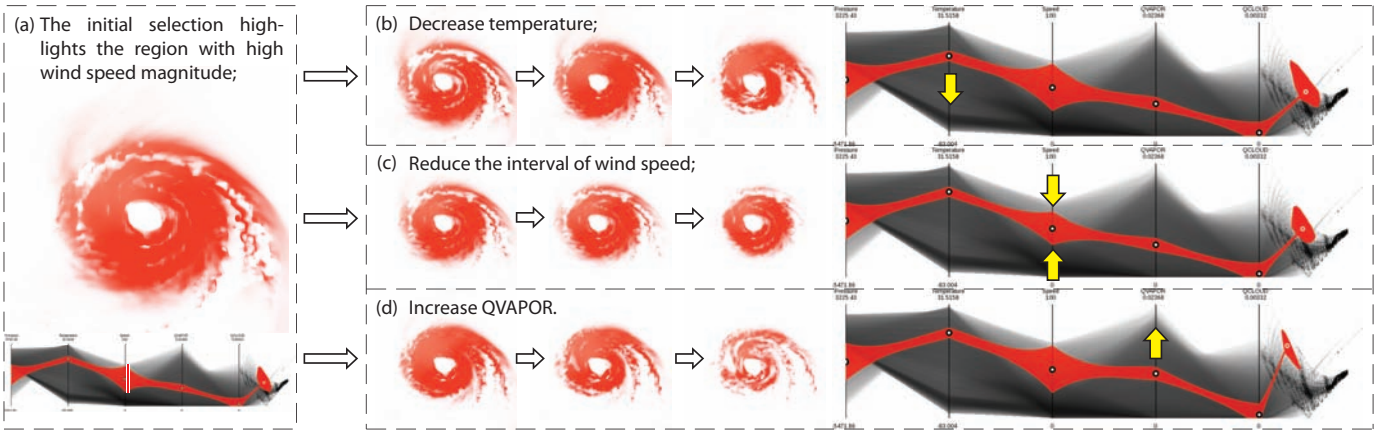


Fig. 14. Interactive exploration of Hurricane Isabel. From the initial selection in (a), the results can be modified by changing the numerical ranges on the PCP. (b)-(d) present the animation sequences during user interaction.

7.2 Ionization Front Instability Simulation Data

The ionization front instability simulation data is from the IEEE Visualization contest 2008 (http://viscontest.sdsc.edu/2008/), which aimed at understanding the ionization front instability. This dataset consists of 12 dimensions, including particle density, gas temperature, speed, vorticity magnitude, and the mass abundance of H, H^+, He, He^+, He^{++}, H^-, H_2, H_2^+ ions. We construct a 12 dimensional TF to generate the visualization results (Figure 15), which bring out some insights into the data. In the purple cluster, where the vorticity is higher, is the region of shock gas. The ingredient of ions are mixed in this region, which is shown in the PCP. This phenomenon is also reflected in the MDS plot (on the right of the PCP), as the purple region covers a large area in the projected space. The green ionization front has medium mass abundance of He and He^+, where there are some transformation process from He to He^+. This region has higher density, medium speed and vorticities. In the central area of the green region, there is a yellow structure, where a much more intense process is happening, as the temperature and the speed are very high. Blue and red regions are transitive features, which provide context of the data. Scientists can conveniently define more insightful features using our TF design interface, in order to test hypotheses, and find meaningful structures in the simulation.

7.3 Turbulence Data

Figure 16 presents the rendering results of a numerical simulation of decaying, compressible, and homogeneous turbulence. The visualization results aim at illustrating the complicated fluid structures. The model of the fluid simulation is dry air, and the initial density and pressure is constant. There are random and uncorrelated sinusoidal velocity perturbations in the periodic cubical volume. The initial RMS Mach number is unity, as the amplitude of the initial power spectrum of velocity fluctuations is chosen. Significant compressible flow and shock waves are generated. The computational mesh used for the simulation is $1,000^3$, and the storage size of each time step is about 12GB.

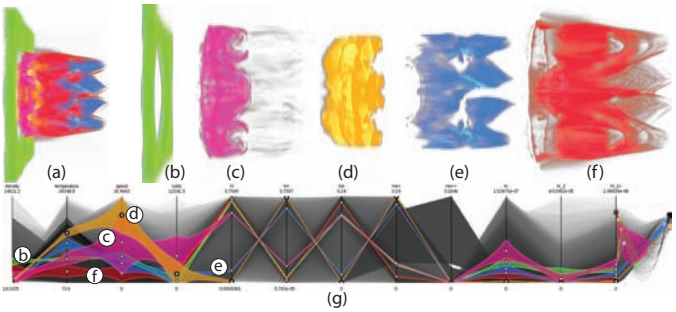


Fig. 15. (a) The rendering result of the ionization front instability simulation data with the multivariate TF specified in (g). (b)-(f), Different components of the data, corresponding to the Gaussian blobs of the TF.

Three of the variables in this multivariate volume visualization case are focused, and they are useful for compressible turbulent flow diagnosis, including the entropy s, the divergence of velocity, $\nabla \cdot \mathbf{U}$, and the vorticity magnitude $|\nabla \times \mathbf{U}|$. The variables contain different structures and features of the fluid simulation. The entropy s, which is a thermodynamic quantity, only changes in response to a dissipative process. As the compressible flow evolving, the shock waves and dissipate kinetic energy are converted into heat, thus increasing the entropy in isolated regions and patches. As the turbulence evolves, the entropy variation develops from the turbulent mixing into surrounding gases. The velocity divergence $\nabla \cdot \mathbf{U}$ measures the compressibility of the flow, which characterizes shock waves and sound. The concentrated regions of velocity jumps tend to be shock waves in the simulation, especially for negative values. Vorticity magnitude $|\nabla \times \mathbf{U}|$, which measures shear in the velocity field, is the main variable in the turbulence evolution. Large values of vorticity can depict both slip surfaces and vortex tubes in the simulation.

As shown in Figure 16, the numerical distributions, as well as the correlations between the three variables are clearly shown in the parallel coordinates. Most of the data points are concentrated in a small range of values. The point cloud in the MDS plot indicates the dissimilarities of the multivariate samples. In the volume rendering result, the red and yellow

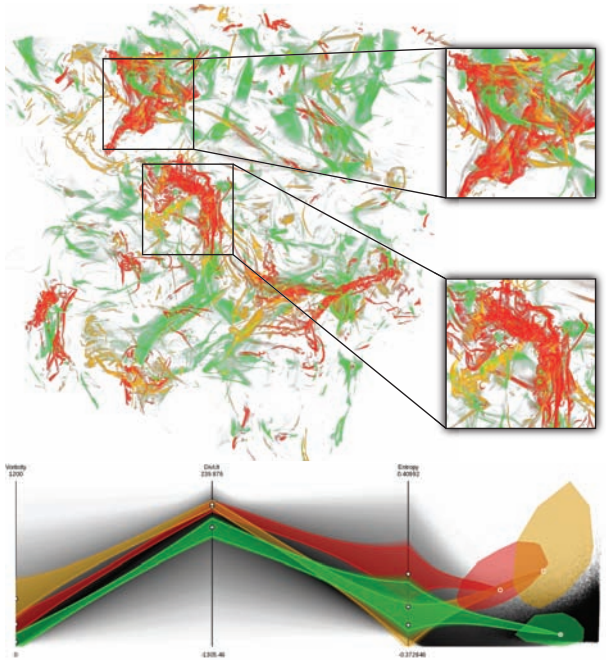


Fig. 16. The rendering result and corresponding multivariate transfer function for a turbulence data set.

components have almost the same distribution on velocity divergence, which reveals the shock waves in the fluid. The geometry structures of the shock waves are clearly shown. In the yellow regions, the vorticity values are high but the entropy values are low. The red regions are complementary to the yellow region with lower values in vorticity but higher in entropy, which indicates significant energy conversion. Unlike red regions, there are also more small vertex tube structures in the yellow regions. Quite different to the regions of shock waves, the green regions present the segmentation information of the fluid structures, since the velocity divergence is relatively low in the field.

8 Conclusions and Future Work

In this paper, we present a multivariate volume visualization system with a novel transfer function design interface. Our design integrates multiple data exploratory methods, including the PCP embedded with MDS plots, and the volume rendering view. The proposed method takes advantage of both a PCP and MDS plots by providing numerical correlation and cluster layout simultaneously. Besides the high performance of the TF generation algorithms and other convenient user interaction techniques, domain experts can quickly make feature selections on any of the components to generate insightful visualization results taking only a few steps, while being aware of the distribution information about the data. Furthermore, we implement our system in parallel computing environments. The volume rendering, MDS plots, and adaptive continuous PCP rendering are accelerated by parallelism, which further accelerates the interactive exploration of larger data sets.

There are a few limitations of our work. First, instead of a real continuous method, a hybrid strategy of continuous and discrete data processing is applied as the hybrid method can keep essential discrete information for clustering and analysis. Continuous dimension projection can be a future improvement, which would allow the data presentation to be continuous and keep more information. Second, our tool relies on manual feature selection. Users need to identify multidimensional features from the user interface, and then highlight the features of interest to explore the data. In our future work, automatic feature detection techniques can be integrated into the systems to facilitate the exploration.

A few more extensions for this work can be developed in the future. First, a systematic evaluation of our system by real users will also be conducted to improve our system. Further more, our system has potential capabilities for large scale temporal data sets. Insightful results could be generated by exploiting some feature tracking and interpolation algorithms. The TF construction can utilize other base functions instead of the Gaussian TF. We believe that effective projection other than Pivot MDS can be integrated into our framework for specific data sets, e.g. LLE or SOM, depending on the data features.

Acknowledgements

The authors would like to thank all the reviewers for their constructive comments to improve the manuscript. Thanks to Min Lu for helping on the performance testing. The authors would also like to thank the institutes (UCAR, SDSC, and LCSE) which make the data available. This work is supported by National Natural Science Foundation of China Project (No. 60903062 and 61170204), 863 Program Project (No. 2010AA012400), Chinese Ministry of Education Key Project No. 109001. This work is also partially supported by the Research Fund for the Doctoral Program of Higher Education of China under. Grant No.200800011004, and the "Strategic Priority Research Program - Climate Change: Carbon Budget and Relevant Issues" of the Chinese Academy of Sciences-Grant No. XDA05040205.

References

[1] H. Akiba and K.-L. Ma. A tri-space visualization interface for analyzing time-varying multivariate volume data. In *Proceedings of EuroVis 2007*, pages 115–122, 2007.

[2] H. Akiba, K.-L. Ma, J. H. Chen, and E. R. Hawkes. Visualizing multivariate volume data from turbulent combustion simulations. *IEEE Computing in Science and Engineering*, 9(2):86–93, 2007.

[3] S. Bachthaler and D. Weiskopf. Efficient and adaptive rendering of 2-d continuous scatterplots. *Comput. Graph. Forum*, 28(3):743 – 750, 2009.

[4] J. Blaas, C. P. Botha, and F. H. Post. Extensions of parallel coordinates for interactive exploration of large multi-timepoint data sets. *IEEE Trans. Vis. Comput. Graph.*, 14(6):1436–1451, 2008.

[5] U. Brandes and C. Pich. Eigensolver methods for progressive multidimensional scaling of large data. In *GD'06: Proceedings of the 14th international conference on Graph drawing*, pages 42–53, 2007.

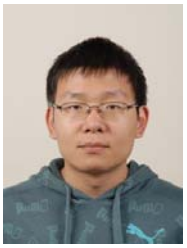
[6] C. Correa and K.-L. Ma. Size-based transfer functions: A new volume exploration technique. *IEEE Trans. Vis. Comput. Graph.*, 14(6):1380–1387, 2008.

[7] C. Correa and K.-L. Ma. The occlusion spectrum for volume classification and visualization. *IEEE Trans. Vis. Comput. Graph.*, 15(6):1465–1472, 2009.

[8] C. Correa and K.-L. Ma. Visibility driven transfer functions. In *Proceedings of IEEE Pacific Visualization 2009*, pages 177–184, 2009.

[9] F. de Moura Pinto and C. Freitas. Design of multi-dimensional transfer functions using dimensional reduction. In *Proceedings of EuroVis 2007*, pages 131–138, 2007.

[10] S. Fang, T. Biddlecome, and M. Tuceryan. Image-based transfer function design for data exploration in volume visualization. In *Proceedings of IEEE Visualization 1998*, pages 319–326, 1998.
[11] H. Guo, H. Xiao, and X. Yuan. Multi-dimensional transfer function design based on flexible dimension projection embedded in parallel coordinates. In *Proceedings of IEEE Pacific Visualization Symposium 2011*, pages 19–26, 2011.
[12] M. Haidacher, D. Patel, S. Bruckner, A. Kanitsar, and E. Gröller. Volume visualization based on statistical transfer-function spaces. In *Proceedings of IEEE Pacific Visualization 2010*, pages 17–24, 2010.
[13] T. He, L. Hong, A. E. Kaufman, and H. Pfister. Generation of transfer functions with stochastic search techniques. In *Proceedings of IEEE Visualization 1996*, pages 227–234, 1996.
[14] J. Heinrich and D. Weiskopf. Continuous parallel coordinates. *IEEE Trans. Vis. Comput. Graph.*, 15(6):1531–1538, 2009.
[15] D. Holten and J. J. V. Wijk. Evaluation of cluster identification performance for different PCP variants. *Comput. Graph. Forum*, 29(3):793–802, 2010.
[16] A. Inselberg. The plane with parallel coordinates. *The Visual Computer*, 1(2):69–91, 1985.
[17] C. Jones, K.-L. Ma, S. Ethier, and W.-L. Lee. An integrated exploration approach to visualizing multivariate particle data. *IEEE Computing in Science and Engineering*, 10(4):20–29, 2008.
[18] G. L. Kindlmann, R. T. Whitaker, T. Tasdizen, and T. Möller. Curvature-based transfer functions for direct volume rendering: Methods and applications. In *Proceedings of IEEE Visualization 2003*, pages 513–520, 2003.
[19] J. Kniss, G. L. Kindlmann, and C. D. Hansen. Multidimensional transfer functions for interactive volume rendering. *IEEE Trans. Vis. Comput. Graph.*, 8(3):270–285, 2002.
[20] J. Kniss, S. Premoze, M. Ikits, A. E. Lefohn, C. D. Hansen, and E. Praun. Gaussian transfer functions for multi-field volume visualization. In *Proceedings of IEEE Visualization 2003*, pages 497–504, 2003.
[21] M. Levoy. Display of surfaces from volume data. *IEEE Comput. Graph. Appl.*, 8(3):29–37, 1988.
[22] K.-L. Ma. Machine learning to boost the next generation of visualization technology. *IEEE Comput. Graph. Appl.*, 27(5):6–9, 2007.
[23] K.-L. Ma, J. S. Painter, C. D. Hansen, and M. F. Krogh. Parallel volume rendering using binary-swap compositing. *IEEE Comput. Graph. Appl.*, 14(4):59–68, 1994.
[24] R. Maciejewski, I. Woo, W. Chen, and D. S. Ebert. Structuring feature space: A non-parametric method for volumetric transfer function generation. *IEEE Trans. Vis. Comput. Graph.*, 15(6):1473–1480, 2009.
[25] K. T. McDonnell and K. Mueller. Illustrative parallel coordinates. *Comput. Graph. Forum*, 27(3):1031–1038, 2008.
[26] D. Patel, M. Haidacher, J.-P. Balabanian, and M. E. Gröller. Moment curves. In *Proceedings of IEEE Pacific Visualization 2009*, pages 201–208, 2009.
[27] F. V. Paulovich, D. Eler, J. Poco, C. Botha, R. Minghim, and L. Nonato. Piece wise laplacian-based projection for interactive data exploration and organization. *Comput. Graph. Forum*, 30(3):1091–1100, 2011.
[28] F. V. Paulovich, C. T. Silva, and L. G. Nonato. Two-phase mapping for projecting massive data sets. *IEEE Trans. Vis. Comput. Graph.*, 16(6):1281–1290, 2010.
[29] T. Peterka, D. Goodell, R. B. Ross, H.-W. Shen, and R. Thakur. A configurable algorithm for parallel image-compositing applications. In *Proceedings of IEEE/ACM Supercomputing 2009 Conference*, pages 1–10, article 4, 2009.
[30] H. Pfister, W. E. Lorensen, C. L. Bajaj, G. L. Kindlmann, W. J. Schroeder, L. S. Avila, K. Martin, R. Machiraju, and J. Lee. The transfer function bake-off. *IEEE Comput. Graph. Appl.*, 21(3):16–22, 2001.
[31] C. Rezk-Salama, M. Keller, and P. Kohlmann. High-level user interfaces for transfer function design with semantics. *IEEE Trans. Vis. Comput. Graph.*, 12(5):1021–1028, 2006.
[32] P. Sereda, A. Vilanova, and F. A. Gerritsen. Automating transfer function design for volume rendering using hierarchical clustering of material boundaries. In *Proceedings of EuroVis 2006*, pages 243–250, 2006.
[33] I. Takanashi, E. B. Lum, K.-L. Ma, and S. Muraki. ISpace: Interactive volume data classification techniques using independent component analysis. In *Proceedings of Pacific Conference on Computer Graphics and Applications 2002*, pages 366–374, 2002.
[34] W. Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 17(4):401–419, 1952.
[35] F.-Y. Tzeng, E. B. Lum, and K.-L. Ma. An intelligent system approach to higher-dimensional classification of volume data. *IEEE Trans. Vis. Comput. Graph.*, 11(3):273–284, 2005.
[36] F.-Y. Tzeng and K.-L. Ma. A cluster-space visual interface for arbitrary dimensional classification of volume data. In *Proceedings of Eurographics/IEEE Symposium on Visualization (VisSym 2004)*, pages 17–24, 338, 2004.
[37] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(4):376 –380, 1991.
[38] Y. Wang, W. Chen, G. Shan, T. Dong, and X. Chi. Volume exploration using ellipsoidal gaussian transfer functions. In *Proceedings of IEEE Pacific Visualization 2010*, pages 25–32, 2010.
[39] P. C. Wong and R. D. Bergeron. Multiresolution multidimensional wavelet brushing. In *Proceedings of IEEE Visualization 1996*, pages 141–148, 1996.
[40] H. Yu, C. Wang, R. W. Grout, J. H. Chen, and K.-L. Ma. In situ visualization for large-scale combustion simulations. *IEEE Comput. Graph. Appl.*, 30(3):45–57, 2010.
[41] H. Yu, C. Wang, and K.-L. Ma. Massively parallel volume rendering using 2-3 swap image compositing. In *Proceedings of IEEE/ACM Supercomputing 2008 Conference*, pages 1–11, article 48, 2008.
[42] X. Yuan, P. Guo, H. Xiao, H. Zhou, and H. Qu. Scattering points in parallel coordinates. *IEEE Trans. Vis. Comput. Graph.*, 15(6):1001–1008, 2009.
[43] X. Yuan, M. X. Nguyen, B. Chen, and D. H. Porter. HDR VolVis: High dynamic range volume visualization. *IEEE Trans. Vis. Comput. Graph.*, 12(4):433–445, 2006.
[44] X. Yuan, H. Xiao, H. Guo, P. Guo, W. Kendall, J. Huang, and Y. Zhang. Scalable multi-variate analytics of seismic and satellite-based observational data. *IEEE Trans. Vis. Comput. Graph.*, 16(6):1413–1420, 2010.
[45] B. Zhang, Y. Ruan, T.-L. Wu, J. Qiu, A. Hughes, and G. Fox. Applying twister to scientific applications. In *CloudCom 10: Proceedings of IEEE Cloud Computing Conference*, pages 25–32, 2010.
[46] X. Zhao and A. Kaufman. Multi-dimensional reduction and transfer function design using parallel coordinates. In *Proceedings of IEEE/EG International Symposium on Volume Graphics 2010*, pages 69–76, 2010.



Hanqi Guo received BS degree in mathematics and applied mathematics from Beijing University of Posts and Telecommunication in 2009. He has been a PhD student on computer science at school of EECS, Peking University since fall 2009. His major research interests include volume visualization, large data visualization and high dimensional data visualization.

He Xiao received BS degree with honor in computer science from Peking University in 2009. He is now a PhD student on computer science at school of EECS, Peking University. His major research interests lie in information visualization and visual analytics, with emphasis on high dimensional data visualization, spatial-temporal data visualization.

Xiaoru Yuan received BS degree in computer science and BA degree in law from Peking University in 1997 and 1998 respectively. In 2005 and 2006, he received MS degree in computer engineering and PhD degree in computer science at University of Minnesota at Twin Cities. He is now a professor at Peking University, in the Laboratory of Machine Perception (MOE). His primary research interests lie in the field of scientific visualization, information visualization and visual analytics with emphasis on large data visualization, high dimensional data visualization, graph visualization and novel visualization user interface.