# Visualization and Exploration of Temporal Trend Relationships in Multivariate Time-Varying Data

Teng-Yok Lee, *Student Member, IEEE* and Han-Wei Shen

**Abstract**—We present a new algorithm to explore and visualize multivariate time-varying data sets. We identify important trend relationships among the variables based on how the values of the variables change over time and how those changes are related to each other in different spatial regions and time intervals. The trend relationships can be used to describe the correlation and causal effects among the different variables. To identify the temporal trends from a local region, we design a new algorithm called SUBDTW to estimate when a trend appears and vanishes in a given time series. Based on the beginning and ending times of the trends, their temporal relationships can be modeled as a state machine representing the trend sequence. Since a scientific data set usually contains millions of data points, we propose an algorithm to extract important trend relationships in linear time complexity. We design novel user interfaces to explore the trend relationships, to visualize their temporal characteristics, and to display their spatial distributions. We use several scientific data sets to test our algorithm and demonstrate its utilities.

**Index Terms**—SUBDTW, trend sequence, trend sequence clustering.

## 1 Introduction

Almost all scientific simulations produce multivariate results. When analyzing a multivariate data set, one important task is to understand the correlation of the variables and their underlying cause-effect relationships. When the data set is time-varying, each variable will exhibit certain temporal trends at different time intervals. By looking at how the trends evolve over time, and how the trends from different variables are related to each other, the scientists can gain better insight into the data and formulate additional hypotheses. Figure 1 shows a simple example, where three temporal trends from wind magnitude, pressure, and temperature, taken from a data point in a hurricane data set, are displayed. It can be seen that in this example, a sudden drop of wind magnitude and pressure happens almost at the same time. On the other hand, the temperature does not seem to be affected although it does arrive at a peak shortly after the dip of wind magnitude and pressure. The combination of these three trends may correspond to an important scientific phenomenon. In fact, in this case they represent the hurricane eye.

Displaying the relationships among important temporal trends provides an opportunity for the scientist to identify correlations across multiple variables and search for salient features from the data set. The relationships associated with a multivariate time-varying data set can be described by the trends' time orders, the durations, and the existence and co-existence of one or multiple trends from the variables. It is also possible to classify the spatial regions based on the trend relationships among different variables. For example, from the data generated by a weather simulation calculated within a duration of 12 months, it can be seen that the solar energy in the northern hemisphere reaches its peak in the summer, followed by an increase in the cloud coverage lasting till the end of year, while an opposite trend order is found in the southern hemisphere. By classifying the existing trend relationships, spatial regions with different trend sequences can be identified.

Most of the existing visualization techniques do not directly display temporal relationships among the variables in multivariate time-varying data sets. Visualizing time-varying data via animations, for example, does not explicitly track the variables' temporal trends. The existing techniques for multivariate data such as parallel coordinates or scatter plots [2] [4] [14] [20] [21] only show the correlation among the scalars, but not necessarily the temporal trends. Instead of considering the data in each time step individually, when analyzing the dynamic behavior of a variable, one should consider the change of value in time. The existing time-series-based algorithms [6] [10] [15] [19] [25] are not explicitly tailored after multivariate data sets. Even though these algorithms could be extended to time series in higher dimensions, they are not designed to extract the temporal sequence of the trends from different variables. Finally, it is more difficult to use the query-based visualization methods [5] [8] [12] [16] [18] if the trends of data are not known in advance.

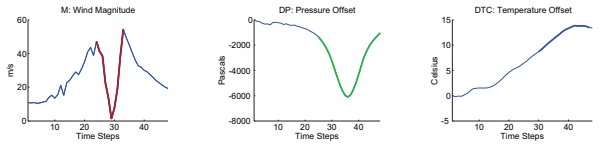
In this paper, we propose a new analysis method for visualizing multivariate time-varying data sets. We allow the user not only to identify and visualize salient temporal trends, but also to model the spatiotemporal relationships of those trends across multiple variables. A new algorithm called *SUBDTW* for comparing and extracting salient temporal trends from the time series defined at a spatial data point is proposed. We also present a method to model and cluster the temporal trends into a collection of trend sequences; the evolution of the trends over time among multiple variables is modeled as a state machine, where the appearance or disappearance of any trend is considered as a state transition in the state machine. From the model of the state machine, a clustering algorithm of linear time complexity for the trend sequences is proposed. This allows the user to have a quick overview of the salient temporal events and their sequences in the data set. To query and visualize the spatiotemporal relationships of different variables, we design several interfaces to allow the user to pick particular trend sequences, and design transfer functions to highlight data of various spatiotemporal properties using volume rendering.

The paper has several unique contributions. It presents a novel algorithm to allow a more detailed visual analysis of spatiotemporal events for time-varying multivariate volume data sets. Compared to the existing metric called Dynamic Time Warping (DTW) that was used in our previous work for univariate scalar data [15], the new and more general algorithm SUBDTW presented in this paper has the same computational complexity as DTW but can detect repeating patterns or patterns in a subset of a time series. In addition, unlike the conventional clustering algorithms such as K-mean or hierarchical clustering which have a complexity quadratic to the input size, our clustering algorithm for trend sequences only has a linear time complexity, and hence allows the user to interact with the time-varying data more easily. By visualizing and exploring the spatiotemporal properties of the data, we provide a new perspective to understand and explore multivariate time-varying data sets.

The structure of this paper is organized as follows. We first re-

• *Teng-Yok Lee and Han-Wei Shen are with The Ohio State University, E-mail: {leeten—hwshen}@cse.ohio-state.edu.*

*Manuscript received 31 March 2009; accepted 27 July 2009; posted online 11 October 2009; mailed on 5 October 2009.*
*For information on obtaining reprints of this article, please send email to: tvcg@computer.org .*




(a) (b) (c)

Fig. 1. Examples of trends from the data set *Isabel*, a hurricane simulation. The trends of interest are plotted in thicker lines with colors red, green and blue, respectively, from (a) - (c). (a): trend $M$ from the wind magnitude, where the valley indicates the appearance of the hurricane eye. (b): trend $DP$ from the pressure. (c): trend $DTC$ from the temperature.

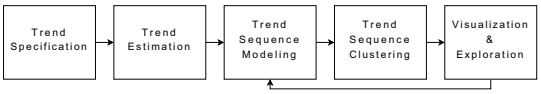



Fig. 2. The data analysis pipeline.

view related work on visualization of time-varying and multivariate data in Section 2, and then describe the specification and detection of temporal trends via SUBDTW in Section 3. The modeling and clustering of trend sequences are discussed in Section 4, and the interfaces for exploring and visualizing spatiotemporal properties of the temporal trends are presented in Section 5. Details about the specification of target trends and the performance of SUBDTW and clustering are proposed in Section 6. Results from case studies using different data sets are listed in Section 7. We discuss the limitations of our work and propose future work in Section 8, and then finally conclude the paper in Section 9.

## 2 Related Work

Visualization of time-varying data has been extensively studied. One conventional method to visualize time-dependent phenomena is through animations. Animations, however, do not extract or track temporal events explicitly. In the literature, there exist several alternatives for visualizing time-varying data. Examples are fusing multiple volumes across time into a single view [24] [26], or considering the time-dependent data as a volume of time series. In medical applications, data points can be classified based on the signals captured over time [6] [10]. The time series data collected at each sample point, called TACs (Time-Activity Curves), can be used as a feature descriptor. Different approaches have been proposed to visualize and analyze TACs for scientific data [15] [19] [25]. The probability distribution or other statistical quantities such as mutual information over time can be used as another important feature of a time-varying data set [1] [2] [22].

The main limitation of the methods above is that only one variable can be considered at a time, which is not sufficient for understanding the correlation among the variables in multivariate time-varying data sets. Previously, methods have been proposed to project or combine multiple variables into a single scalar to highlight the correlation [3] [7] [9]. In the literature of information visualization research, ThemeRiver is a technique to visualize the populations of multiple time series together [11]. The parallel coordinates plot [13] is a popular visualization technique to understand static multivariate data sets. Several extensions have been proposed for time-varying cases [2] [4] [14] [20] [21]. One common issue of these techniques in both scientific and information visualization is that they are mainly designed to understand the correlation among the quantities, not among the time-dependent trends.

To visualize the correlation among the variables or events over time, one choice is to use the query-based visualization methods. After the desired events over time and their order are specified in a query, the regions and time steps that satisfy the query are highlighted. The query can be specified via mathematical expressions or regular expressions [8] [16] [18], or via graphical user interfaces like TimeBox [12] or Pattern Finder [5]. The main difference between these works and ours is that our method does not require a known temporal order among the events to be specified. Instead, our method can automatically detect all salient sequences of the events, which can be useful for exploring unknown phenomena.

It it noteworthy that our previous work in [15] utilizes the distance metric DTW to aggregate a time-varying scalar field into a distance field for a given time-varying feature. This distance field is called TAC-based distance field, which can be rendered to visualize the spatiotemporal behavior of the feature. In this paper, we extend our previous work substantially with three major differences. First, the well-known DTW distance metric is extended to a new and more robust SUBDTW distance metric. Second, this study extends the data domain from univariate to multivariate. Third, while our previous work only covers the rendering of TAC based distance fields, this papers provides a more complete visualization system to explore and analyze time-varying data sets.

## 3 Trend Specification and Estimation

Our data analysis pipeline consists of the following stages, also shown in Figure 2. At the beginning, the user specifies the trends of interest, called *target trends*, to start the visualization process. The target trends can be specified by the users based on their domain knowledge, or extracted from the data using a clustering algorithm. Next, for each data point, our system applies SUBDTW to estimate whether and when the specified trends occur. After the estimation is completed, the time steps of all trends at the data point form a trend sequence, which allows us to model the temporal relationships among the trends. The trend sequences of all data points are clustered to extract the most salient trend sequences in the data set. Finally, the user can browse and visualize the salient trend sequences to understand their spatiotemporal behavior. The user can change the parameters to generate another set of trend sequences on the fly, indicated by the loop shown in Figure 2. In the following, we describe each stage in detail.

### 3.1 Trend Specification

We denote a multivariate time-varying volume data set with $m$ variables and $n$ time steps as $f_i(x,y,z,t) \in \mathscr{D}_i$, where $(x,y,z) = \mathbf{x}$ is the position of the voxel, $i = 1 \ldots m$, $t = 1 \ldots n$, and $\mathscr{D}_i$ is the domain for the $i$-th variable. Each voxel, hereafter called a data point, has $m$ time series, one from each variable. A time series of the $i$-th variable on a data point $\mathbf{x} = (x,y,z)$ can also be denoted as $f_{i,\mathbf{x}}[t] = f_i(x,y,z,t), t = 1 \ldots n$.

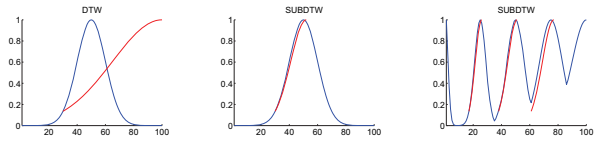
Using time series as feature descriptors is common in many medical and scientific applications. Figure 1 (a), for instance, shows the wind magnitude collected at a data point in a hurricane simulation. We can see that the wind magnitude quickly drops to almost zero as the hurricane eye is passing through.

From the multivariate time series, it is possible to identify one or multiple important temporal trends associated with each variable. The three trends in Figure 1 are extracted from wind magnitude, pressure and temperature of a hurricane data set. We denote the trends of the $i$-th variable as $p_j[t] \in \mathscr{D}_i, t = 1 \ldots n_j$, $j = 1 \ldots k$ where $k$ is the number of the trends of interest, and $n_j$ is the number of time steps in trend $j$. Note that a trend itself is a time series, although its length, i.e., the number of time steps, can be different from the total number of time steps in the data set.

Note that when multiple target trends are applied on a single variable, each target trend will be identified individually. Given a set of target trends, if the pattern of a target trend is contained in other target trends, it might create redundant or ambiguous result in later stages. The influence of such a case requires further study to verify.

### 3.2 SUBDTW

An important step of our algorithm is to identify important temporal trends from the data set so that the characteristics of these trends in space and time can be visualized. Given a target trend $p[t]$ of the $i$-th variable, we search where and when it occurs in the data by comparing it with the time series $f_{i,\mathbf{x}}[t]$ at each data point $\mathbf{x}$. We say that a data point $\mathbf{x}$ exhibits the target trend $p[t]$ if the distance or dissimilarity $d$




(a) (b) (c)

Fig. 3. A comparison between DTW and SUBDTW. (a) and (b): search for an increasing function (red) in a time series (blue). The warping via DTW and SUBDTW are plotted in (a) and (b), respectively. While DTW inaccurately warps the pattern to the whole time series, SUBDTW can detect the sub time series that best matches the pattern. (c): detection of the repeating occurrence of the increasing function in another time series via SUBDTW.

between the target trend and a sub time series at $\mathbf{x}$ within a time interval $\mathscr{T}$ is smaller than a given threshold $\delta$.

To calculate the distance and the time interval within which the sub time series of a data point matches the target trend, a distance metric between two time series will be needed. Considering that two time series may contain a similar temporal trend but with different starting time and length, the distance metric needs to be much more robust than simple metrics such as $L_1$ and $L_2$ norms used in previous works [6] [10] . The distance metric should also consider the case when the trend is stretched or compressed non-uniformly in the time series, which cannot be detected by maximum cross correlation [6].

Dynamic Time Warping (DTW) is a metric to measure the distance between two time series. It is computed using dynamic programming to warp one time series non-linearly to match the other with a least distortion. DTW is widely used in speech recognition and data mining, and was used in our previous work to perform spatiotemporal analysis for scientific applications [15]. DTW computes the smallest distance after the warping between two time series $p[1\dots n_p]$ and $q[1\dots n_q]$ based on the following constraints:

1. the first and last time steps in one series are always mapped to the first and last time steps of the other series;
2. the warping should preserve the original order of time steps in the series;
3. two adjacent time steps in one series cannot be mapped to non-adjacent locations in the other series.

The smallest distance can be computed with the formulation in Equation 1:
For $i = 1\dots n_p$, $j = 1\dots n_q$

$$D[i,j] = dist(p[i],q[j]) + \begin{cases} 0 & i=1 \& j=1 \\ D[i,j-1] & i=1 \& j>1 \\ D[i-1,j] & i>1 \& j=1 \\ D_2(i,j) & \text{otherwise} \end{cases} \quad (1)$$

where

$$D_2(i,j) = min\{D[i-1,j], D[i-1,j-1], D[i,j-1]\} \quad (2)$$

The optimal distance after the warping is $D[n_p,n_q]$.

Since our goal is to match the target trend with any sub time series of a data point, DTW does not suit our need. This is because DTW cannot match one time series to a subset of the other time series. Figure 3 (a) shows an example of matching two time series (blue and red). The blue time series reaches its peak at time step 50 and then drops, while the red series is a monotonically increasing function. DTW will fail to match the red series to the first half of the blue series.

The fundamental reason why DTW cannot match sub time series is due to the first constraint above, which forces the algorithm to consider the entire sequence. By relaxing the first constraint, DTW can be modified to generate a best match between one time series and a subset of the other. Below we propose a modified algorithm, called *SUBDTW*, that satisfies this goal.

Assuming that a target trend is specified as a time series $p[1\dots n_p]$, and the time series to be searched is $q[1\dots n_q]$, we can modify the recursive equations as in Equation 3 to compute the distance between two time series:
For $i = 1\dots n_p$, $j = 1\dots n_q$,

$$D[i,j] = dist(p[i],q[j]) + \begin{cases} 0 & i=1 \& j=1 \\ 0 & i=1 \& j>1 \\ D[i-1,j] & i>1 \& j=1 \\ D_2(i,j) & \text{otherwise} \end{cases} \quad (3)$$

The main change from Equation 1 to Equation 3 is in the case when $i = 1$ and $j > 1$, where Equation 3 does not consider the neighboring cell $D[i, j-1]$. Once $D[i,j]$ for $i = 1\dots n_p$ and $j = 1\dots n_q$ have been obtained, the smallest distance between the pattern $p$ and the sub time series in $q$ is $\min_{j=1\dots n_q} D[n_p,j]$. The time step $j' = \text{argmin}_{j=1\dots n_q} D[n_p,j]$ is the ending time of the target trend $p$ on the time series $q$. The beginning time of the trend can be obtained by backtracking the table $D$ from $(n_p,j')$ to the first cell with a row index of 1. Assume that the index of this cell is $D[1,j'']$, the time step $j''$ is then the beginning time step of the trend on the time series.

The main benefit of our SUBDTW algorithm is that it can match a pattern with a sub time series, which cannot be done by DTW. Besides, the time complexity of SUBDTW is the same as DTW. Moreover, SUBDTW can be applied to detect repeating appearances of a pattern, as shown in Figure 3 (c) where the increasing parts in a time series are detected by SUBDTW. To detect a repeating pattern, the local minima along the sequence $D[n_p,1\dots n_q]$ will be the ending time steps of all appearances. The corresponding beginning time step of each ending time step can be then backtracked.

One drawback of DTW and SUBDTW is the performance since their time complexities are quadratic to the numbers of time steps. More detailed analysis about the performance is discussed in Section 6.2.

## 4 Trend Analysis

After SUBDTW has been applied to search for the target trends in the data set, we can identify the order of these trends appearing at each data point and how the trends from different variables overlap with one another in time. We call this information *trend sequences*. We perform clustering to group the trend sequences from all the data points to extract salient trend sequences for a more detailed visual exploration. Below we describe our trend analysis process in detail.

### 4.1 Trend Sequence Modeling

To model the trend sequences, two types of information are considered. One is a sequence of time points, each of which represents the beginning or ending time of any trend. Assuming there are a total of $k_{\mathbf{x}}$ trends detected from the time series at a data point $\mathbf{x}$, there can be $n_{\mathbf{x}}$ unique time points, denoted as $\{t_{\mathbf{x},1}\dots t_{\mathbf{x},n_{\mathbf{x}}}\}$, to be considered. It is noteworthy that $n_{\mathbf{x}}$ can be different from $2\times k_{\mathbf{x}}$ since multiple trends may begin or end at the same time. In addition, one trend may appear multiple times at a data point.

The other information to consider when sorting out the trend sequences is how the trends from different variables overlap in time. For a data point, the combination of several trends that occur in the same time interval is defined as a *state*. If the total number of the target trends is $k$, there will be at most $2^k$ possible combinations of trends for the data point at any instant in time because each trend can happen at that time point or not. A data point $\mathbf{x}$ starts from an initial state $s_0$ when none of the target trends appears. When time is at $t_{\mathbf{x},1}$, the trend sequence enters the next state $s_{\mathbf{x},1}$, in which one or more trends start to occur and will last between time steps $t_{\mathbf{x},1}$ and $t_{\mathbf{x},2}$. When time comes to $t_{\mathbf{x},2}$, some trends stop and/or some other trends start, so the system enters the next state $s_{\mathbf{x},2}$, and so on. In other words, the state $s_{\mathbf{x},i}$ contains the trends that last between time steps $t_{\mathbf{x},i}$ and $t_{\mathbf{x},i+1}$ for

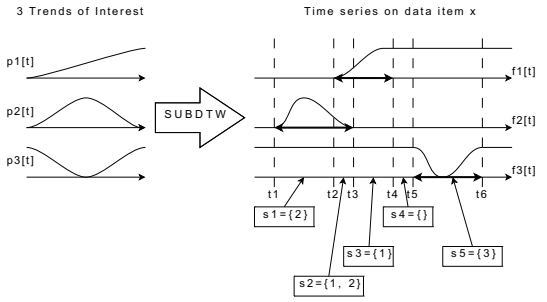



Fig. 4. Trend sequence. Given three trends of interest p1[t], p2[t] and p3[t], once the trend detection via SUBDTW is completed, the beginning and ending times of the three trends $t1/t3$, $t2/t4$, and $t5/t6$, respectively, are detected. Since there are 6 unique time steps, 5 states are created, each of which is a set of indices to indicate the coexistence of trends in the time intervals. The trend sequence is thus $\{t_1, s_1 = \{2\}, t_2, s_2 = \{1,2\}, t_3, s_3 = \{1\}, t_4, s_4 = \{\}, t_5, s_5 = \{3\}, t_6\}$. Note that the three target trends are for different variables, and thus the trend p1 won't be treated as a sub time series in p2 and p3.

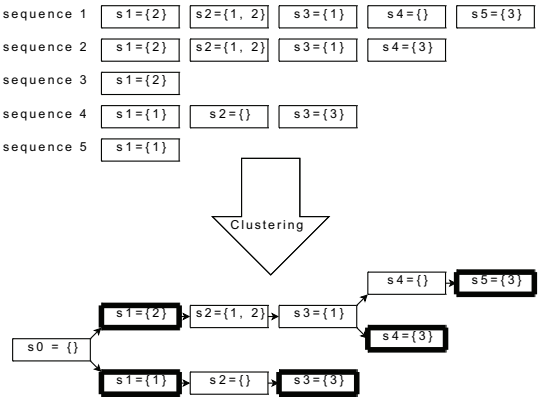



Fig. 5. Trend sequence clustering. Five state sequences are given here. The clustering will group the given state sequences into a tree. In this tree, the state sequences with a common prefix will be placed under a subtree. Each node with a thicker border represents the ending of a state sequence.

$i = 1 \ldots n_{\mathbf{x}} - 1$. The state transition will stop when the last time step $t_{\mathbf{x},n_{\mathbf{x}}}$ is reached.

To summarize, the trend sequence at a data point $\mathbf{x}$ can be described by a sequence of time points $\{t_{\mathbf{x},1} \ldots t_{\mathbf{x},n_{\mathbf{x}}}\}$ which indicates the beginning or ending of any trends, and a sequence of states $\{s_{\mathbf{x},1} \ldots s_{\mathbf{x},n_{\mathbf{x}-1}}\}$ which represents the combination of the trends at the time points in the time sequence. Therefore, the trend sequence can also be denoted as a sequence of time and state pairs $\{t_{\mathbf{x},1}, s_{\mathbf{x},1}, t_{\mathbf{x},2}, s_{\mathbf{x},2} \ldots t_{\mathbf{x},n_{\mathbf{x}}-1}, s_{\mathbf{x},n_{\mathbf{x}}-1}, t_{\mathbf{x},n_{\mathbf{x}}}\}$. Figure 4 shows an example of the trend sequence.

### 4.2 Trend Sequences Clustering

When the trend sequence at every data point has been obtained, clustering can be used to identify some common sequences of the trends in the data set. A straightforward way of clustering is hierarchical clustering, where all pairs of trend sequences are compared and grouped in a bottom-up manner. The time complexity of this step is quadratic to the number of data points, which is too slow for large scale scientific data.

To address the issue, we devise a new algorithm to cluster the trend sequences. The basic idea is to merge the state sequences of individual data points one by one into a global state transition diagram represented as a tree. The diagram starts with only one node $s_0$ as the root, which represents no occurrence of any trends. As each data point's state sequence is merged into the diagram, new branches are created. Each path from the root to the leaf in the tree represents a complete state sequence. State sequences that start with a common subsequence will share a portion of a path from the root before their later subsequences make it necessary to branch out. We describe below the detail of merging one state sequence $\{s_{\mathbf{x},1} \ldots s_{\mathbf{x},n_{\mathbf{x}}}\}$ into the diagram.

Given an input sequence, starting from the root of the diagram, we first check whether there exists an edge from the root to a state that is equal to the first state in the input sequence, $s_{\mathbf{x},1}$. If such an edge exists, we follow the edge and continue to match the second state $s_{\mathbf{x},2}$ of the sequence. Otherwise, a new edge representing $s_{\mathbf{x},1}$ connecting to the root is created. We repeat this process by scanning through all the states in the input sequence, during which create or follow edges as described until the last state in the input sequence is reached, where the corresponding node in the tree is called a *terminal node*. We keep a counter at each terminal node and increase the counter by one at the terminal node every time when an input sequence ends there. This counter is to record how many data points have their state sequences following the same path from the root to this terminal node, i.e., have the same trend sequence.

Once all state sequences have been merged to the diagram, the paths from the root $s_0$ to all terminal nodes represent all possible state sequences in the data set. Note that there exists no loop in the state transition diagram since it is a tree. The tree places the state sequences with a common prefix under the same subtree. The depth of the subtree depends on the length of the common prefix. It should be noted that an internal node in the tree can be a terminal node. Figure 5 is an example of the tree created from 5 state sequences.

Regarding the computational speed, since each data point's state sequence is scanned only once, the time complexity to merge all state sequences is linear to the number of data points. Because our clustering algorithm is efficient, the user can change some parameters and re-cluster the trend sequences on the fly. One important parameter is the threshold of distance $\delta_j$, which decides whether a target trend $p_j$ exists in a time series. The user can tune the threshold and quickly obtain the new clustering result. Different filtering can also be applied to extract salient trend sequences in the data set. One is based on the occurrence of each trend sequence, i.e., how many data points have the same trend sequence. We can either select the most frequently occurring trend sequences or discard the sequences whose occurrences are smaller than a threshold. Another choice is to select based on the number of states in the trend sequences and ignore those that are too simple or complicated.

## 5 Visualization

Once salient trend sequences have been extracted, the user can begin to visualize the spatial and temporal relationships of the trends from the variables. The knowledge of temporal trends in the data can be used to assist the design of transfer functions for volume rendering. In the following, we present our methods in detail.

### 5.1 Trend Sequence Representation

One way to visualize the trend sequences is to show the state diagram mentioned above. However, showing the state diagram does not give the user an intuitive way to understand the time progression of those trends. Therefore, a more effective visualization method is called for.

We visualize each trend sequence as a 2D $n \times k$ image, where $k$ is the number of trends and $n$ is the number of states in the sequence. The $i-$th column in the image represents the $i-$th state, and the $j$-th row represents the $j$-th trends. Each trend $p_j$ is associated with a color $c_j$. For a pixel with an index $(i, j)$, if the $j-$th trend is included in the $i-$th state, this pixel is then colored by $c_j$; otherwise, this pixel is left blank.

An example is shown in Figure 6, where the time series used are the same as in Figure 4. The colors for the three trends are red, green and cyan, respectively. From the image, the temporal order among the three trends can be clearly seen. It can also be observed that there is

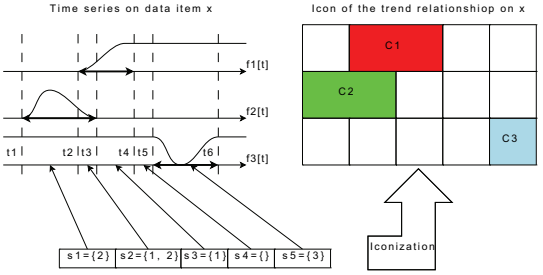



Fig. 6. An example illustrating how a trend sequence consisting of 5 states is transformed to a 2D image with $5 \times 3$ pixels.

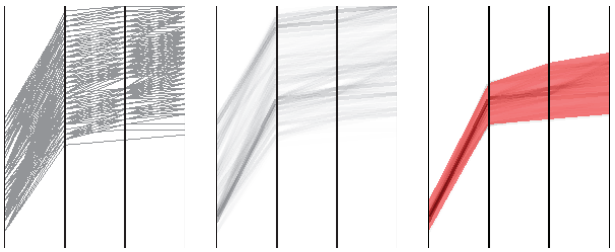



Fig. 7. Parallel coordinates plots indicating the time intervals for the trends. (a): the plot without alpha accumulation and the width of the line segment is 1 pixel. (b): the plot with alpha accumulation (the opacity is 0.1) and wider line segments (the width is 4 pixels), which contains two visual clusters. (c): parallel coordinates tunnel. To design a transfer function, the user specifies time intervals on the axes to define a tunnel to enclose the data points of interest. A tunnel colored in red is shown in subfigure (c), where the bottom cluster in subfigure (b) is selected.

an overlap in time between the first two trends but not between the last trend and any other trends.

### 5.2 Parallel Coordinates for Trend Sequences

While multiple data points can have the same trend sequence, trends in different spatial regions may start at different times with different durations. The above method for visualizing trend sequences displays the temporal order of trends but not the specific time, which can be important for understanding the cause-effect relationship between the different temporal events and the durations of the events. To display the detailed time information for all the data points that share the same trend sequence, parallel coordinates with $n+1$ vertical axes are used for a trend sequence with $n$ distinct states, considering the beginning and ending times of the states. In our plot, one axis is used for each state to represent the time when the trend sequence enters/leaves that state. We then plot every data point's trend sequence as a polyline in the parallel coordinates. An example of the parallel coordinates plot for a trend sequence with 3 states is shown in Figure 7 (a). The slope of the polyline segment indicates the duration of the corresponding state, and the visual clusters in the parallel coordinates plot further separate the data points into different categories.

In our current implementation, each line segment is drawn with an opacity, controlled by the user. Since many data points may enter the same state in the trend sequence at the same time, many line segments in the plot may overlap. By blending the line segments together, the time intervals and the states that have high occurrences will be visually highlighted. The user can also control the widths of the line segments to emphasize the correlation among nearby line segments. Figure 7 (b) shows the modified plot from Figure 7 (a) with the combined effect of blending and using a larger line width, which visually presents the polylines into two groups. More complicated techniques, like those operations proposed in Illustrative Parallel Coordinates [17], can be applied to create more intuitive and effective visualization.

### 5.3 Trend-sequence-based Transfer Function Design

Based on the visualization methods described in the previous sections, we create a user interface to design transfer functions for volume rendering. The goal of the transfer functions is allowing the user to visualize regions that have interesting trend sequences. In the interface, the user first selects a trend sequence of interest by clicking on its representative image that we describe in Section 5.1. From the parallel coordinates plot of the selected trend sequence, the user can choose groups of polylines that are of interest, and render them with different visual attributes to highlight their spatial locations using volume rendering.

To choose a polyline group, our interface lets the user specify a time interval on each axis in the parallel coordinates. Assuming this trend sequence contains $n$ states $\{s_1 \dots s_n\}$ and the selected time interval on the $j$-th axes is denoted as $\{t_j^0, t_j^1\}$, the selected time intervals $\{\{t_1^0, t_1^1\} \dots \{t_{n+1}^0, t_{n+1}^1\}\}$ will form a tunnel across the parallel coordinates axes, as the red regions illustrated in Figure 7 (c). Polylines that fall within the tunnel are selected. The trend sequence $\{s_1 \dots s_n\}$ and the selected time intervals $\{\{t_1^0, t_1^1\} \dots \{t_{n+1}^0, t_{n+1}^1\}\}$ are added as an entry to the transfer function. For each tunnel, two colors $c^0$ and $c^1$ will be specified, from which the colors for polylines within the tunnel will be interpolated. Two examples of the transfer function entries are presented in Figures 10 (a) and (b).

During the volume rendering process, each voxel $\mathbf{x}$ looks up the entries in the transfer function. If this voxel's trend sequence $\{s_{\mathbf{x},1} \dots s_{\mathbf{x},n_\mathbf{x}}\}$ matches a sequence in one of the entries, and its time sequence $\{t_{\mathbf{x},1} \dots t_{\mathbf{x},n_\mathbf{x}+1}\}$ falls in the tunnel, a color $c_\mathbf{x}$ and an opacity $\alpha_\mathbf{x}$ are computed for this voxel. Otherwise this voxel will be transparent.

The color $c_\mathbf{x}$ is interpolated from the colors $c^0$ and $c^1$ assigned to this entry. The color transition can indicate the transition of time for the region that has the same state sequence. The color $c_\mathbf{x}$ is computed as Equation 4:

$$c_\mathbf{x} = (c^0 \omega^1 + c^1 \omega^0)/(\omega^0 + \omega^1) \tag{4}$$

where

$$\omega^k = \sqrt{\sum_{j=1}^{n_\mathbf{x}+1} (t_{\mathbf{x},j} - t_j^k)^2}, \; k = 1 \text{ or } 2. \tag{5}$$

The color $c_\mathbf{x}$ is a linear interpolation of the two colors $c^0$ and $c^1$. The weights $\omega^0$ and $\omega^1$ are decided based on the Euclidean distances from a vector $(t_{\mathbf{x},1} \dots t_{\mathbf{x},n_\mathbf{x}+1})^\mathbf{T}$ to the two bounds of the tunnel, $(t_1^0 \dots t_{n_\mathbf{x}+1}^0)^\mathbf{T}$ and $(t_1^1 \dots t_{n_\mathbf{x}+1}^1)^\mathbf{T}$.

The opacity of each voxel is modulated based on how similar the voxel's temporal trends are to the target trends. Assuming there are $k_\mathbf{x}$ trends in the target trend sequence, the smallest distance from the voxel's time series to the $j-$th target trend is $d_j$, and the cutoff threshold for having enough similarity is $\delta_j$, $j = 1 \dots k_\mathbf{x}$, the opacity $\alpha_\mathbf{x}$ for the voxel is then calculated as in Equation 6, which attenuates the voxel opacity when the distance between its time series and the target trends are larger.

$$\alpha_\mathbf{x} = \exp(-\sum_{j=1}^{k_\mathbf{x}} d_j^2 / \sum_{j=1}^{k_\mathbf{x}} \delta_j^2) \tag{6}$$

Figure 10 (c) and Figure 12 (d) are the resulting images from our volume renderer. We explain the visualization of this data set in more detail in Section 7.

## 6 Implementation

This section provides some more implementation details about our system, including the user interfaces for target trend specification, and the performance analysis of SUBDTW and trend sequence clustering. Our system was implemented on a machine with an Intel Core 2 Duo 6700 processor, 3GB system memory, and an nVidia GeForce GTX 280 graphics adapter.

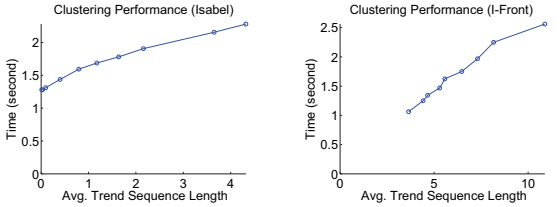



Fig. 8. Performance of the clustering algorithm for out testing data set *Isabel* and *I-Front*.

Table 1. Performance of SUBDTW on the data sets *Isabel* (with $250 \times 250 \times 30$ data points in 48 time steps) and *I-Front* (with $180 \times 62 \times 62$ data points in 200 time steps). The timings are in seconds.

| Data set | *Isabel* | | | *I-Front* | | |
|---|---|---|---|---|---|---|
| | Trend | Length | Time | Trend | Length | Time |
| | *M* | 10 | 59.94 | *curl* | 36 | 99.67 |
| | *DTC* | 18 | 71.81 | $H+$ | 56 | 122.84 |
| | *DP* | 24 | 79.17 | $H_2$ | 61 | 134.64 |
| Total | | | 210.92 | | | 357.15 |

### 6.1 Target Trend Specification

In our current system, two user interfaces are provided to specify the target trends. One interface is similar to the spreadsheet layout proposed by Woodring and Shen [25], which utilizes K-mean clustering to extract and display salient trends. During a preprocessing stage, we apply $K$-mean clustering to group the time series on all data points. The mean time series of all groups are then listed on the screen, where the user will browse and select the time series with interesting patterns as the target trends. To reduce the computational cost, we used the Euclidean distance metric for the K-mean clustering process. We also display a larger number of clusters, for example 30, in order to capture as many salient patterns as possible. Even though these may exist redundance in the extracted patterns since the Euclidean distance metric does not consider the time shift of patterns, the user can pick any one of the redundant patterns as the target trends. Alternatively, another user interface that we provide is to let the user pick a spatial point on the screen using mouse pointer. The time series associated with this spatial point can be then added to the set of target trends.

Once the target trends have been specified, the user can perform some further adjustment. For instance, the user can select only part of the target trends with the most salient or interesting pattern, or scale or shift the scalar range of the target trends.

### 6.2 Performance

The performance of SUBDTW and trend sequence clustering depends on several factors. While both depend on the numbers of target trends, data points and the time steps, the time complexity of SUBDTW is also affected by the length of the target trends. Therefore, the computation of SUBDTW is quadratic to the number of time steps and thus can be the performance bottleneck. For instance, the computation of SUBDTW of the three target trends in Figures 1 took 210.92 seconds for the data set *Isabel*, as shown in Table 1, while the trend sequence clustering took less than 2.17 seconds as shown in Figure 8.

As mentioned earlier, the time complexity of the clustering algorithm is linear to the number of data points. More precisely, in addition to the number of data points, the computational speed also relies on the average length of the trend sequence per data point, i.e., the number of states in the sequence, which in turn depends on the distance thresholds $\delta_j$. Figure 8 shows the relationship between the computation time and the average number of the states. We used different thresholds to control the number of states from the data points to conduct our experiments. Figure 8 empirically demonstrates that the performance linearly depends on the average number of states.

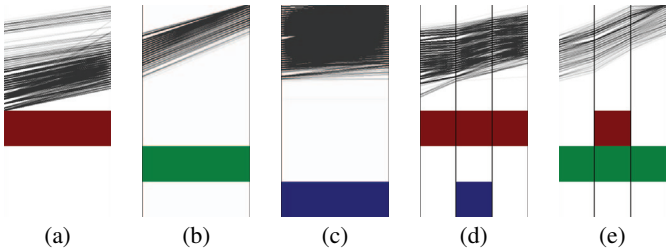



Fig. 9. The salient trend states and their time steps for the trends in Figure 1 from the data set *Isabel*.

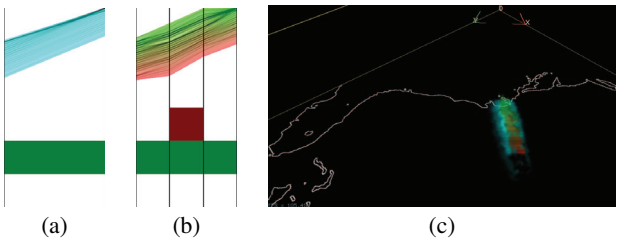



Fig. 10. The transfer functions and volume rendered images for the data set *Isabel*. (a): the transfer function for the trend sequence in Figure 9 (b), which assigns a cyan color. (b): the rendered image when the transfer function in (a) is considered. (c): the transfer function for the trend sequence in Figure 9 (e), which assigns the color from red to green. (d): the rendered image when both transfer function entries in (a) and (c) are considered.

## 7 Case Studies

### 7.1 Hurricane

Our first case study uses the data set *Isabel*, the benchmark for IEEE 2004 Visualization Design Contest. This data set was generated from National Center for Atmospheric Research (NCAR) to simulate Hurricane Isabel, an intense tropical weather system that occurred in September, 2003, over the west Atlantic region. To reduce the storage requirement, a downsampled version of the data at a resolution of $250 \times 250 \times 30$ voxels per time step was used to run out tests. The variables used in our case study are the wind magnitude, pressure, and temperature. Since we were more interested in the variables' temporal trends during the simulation other than their absolute scales, we aligned the time series by subtracting the value of its first time step from each time series.

One important property of a hurricane is the path and structure of the hurricane eye. The hurricane eye can influence the wind magnitude and pressure. The wind magnitude will increase before the arrival of the hurricane eye, but will quickly drop when the hurricane eye enters the region. Then, the magnitude will start to increase back to the original level and then decrease as the hurricane eye moves away. Similarly, when the hurricane eye enters a region, the pressure will quickly drop, and then will return to its normal level as the hurricane eye leaves. The time series for the wind magnitude and pressure over time around the hurricane eye, hereafter denoted as time series $M$ and $DP$, are used as target trends, shown as the thicker line segments in Figure 1 (a) and (b), respectively. Meanwhile, to know when and where the temperature increases to the highest scale in the hurricane, we use a function plotted as a thick line segments in Figure 1 (c) as the target time series, denoted as $DTC$.

In our analysis, we first compared the three target trends with the time series in the data set using SUBDTW. The distance threshold for each trend was empirically set to 0.2 multiplied by the maximal distances from all the data points to the trend. Then, we clustered the trend sequences from those points that exhibit at least one trend. Only the non-empty trend sequences whose occurrences exceed 1% of the total trend sequences were retained. The five salient trend sequences and the time intervals for the states in the sequences are shown in Fig-

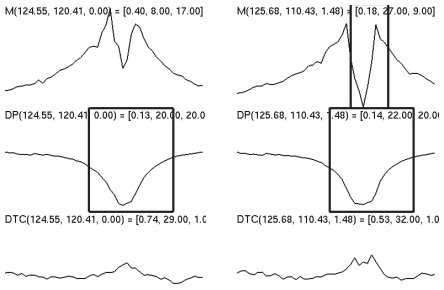



Fig. 11. Two sets of sampled time series. (a) and (b) are sampled from the trend sequences in Figure 9 (b) and (e), respectively.

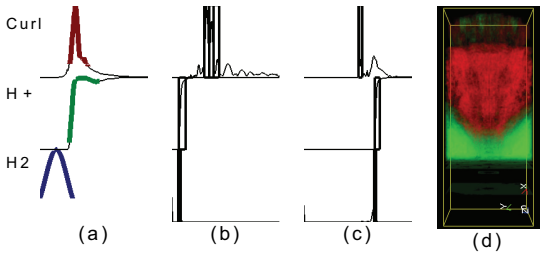



Fig. 12. Case study for the Ionization Front Instability. (a): the target trends. (b): the case where the two elements $H+$ and $H_2$ lead the target trend for the curl field. (c): the case where the target trend for the curl field leads the two elements. The bounding boxes in (b) and (c) represent the detected trends. (d): the volume rendered image, generated by assigning the cases in (b) and (c) as red and green colors, respectively.

ure 9. In each subfigure, the bottom image represents the trend sequence, where the three target trends $M$, $DP$ and $DTC$ are drawn in the color of red, green and blue, respectively, and the top parallel coordinates plot indicates the time progression of the states in the trend sequence.

In our case study, one query we had was to verify whether trends $M$ and $DP$ simultaneously occur, and whether they are related to the increase of temperature. From Figures 9 (a) and (b), it can be seen that the trends $M$ and $DP$ do not always occur simultaneously in the region that contains the hurricane eye since there is no overlap of red and green in both trend sequences. Figure 9 (c) indicates that the temperature can increase in regions that do not contain the hurricane eye. Figures 9 (d) and (e) show the trend sequences that contain more than two coexisting trends, all of which include trend $M$.

Another query in our study was to explore the regions where trend $DP$ occurs. We selected the trend sequences in Figures 9 (b) and (e), and designed a transfer function with two entries, as shown in Figure 10 (a) and (b) to visualize its spatial distribution. The color for the trend sequence in Figure 9 (b) was assigned as cyan, as shown in Figure 10 (a). We specified another transfer function entry as shown in Figure 10 (b), which assigns the trend sequence in Figure 9 (e) with colors from red to green. From the volume rendered image shown in Figure 10 (c), the cyan region encloses the path of the hurricane eye, where the color transition in the area indicates the path of the hurricane eye. From Figure 10 (c), we can observe that when the hurricane eye passes through, the pressure in the region surrounding the path will drop and then increase, but only the region on the path of the hurricane eye will see the change of wind magnitude in the same way as in trend $M$, a sudden drop of wind magnitude.

To verify this observation, we sampled these data points that exhibit the two trend sequences. Figures 11 (a) and (b) represent the sampled time series from the trend sequences in Figures 9 (b) and (e), respectively. The rows from top to bottom are the time series for the magnitude, pressure and temperature, respectively, at a data point. In each time series subfigure, a rectangle is drawn to enclose a sub time series where the distance between the sub time series and the target trend of the same variable is smaller than the distance threshold. In Figure 11 (a), the wind magnitude in that time series does not drop as much as in the trend $M$, and temperature time series looks flatter than trend $DTC$. Meanwhile, in the time series shown in Figure 11 (b), the drop of wind magnitude matches better with trend $M$. Although the temperature increases more than that of the time series shown in Figure 11 (a), the increase still cannot match trend $DTC$.

### 7.2 Ionization Front Instability

In our second case study, we used the benchmark for IEEE 2008 Visualization Design Contest [23]. The purpose of this data set is to explore the influence of ionization front (I-Front) instabilities to the formation of early universe. We will call this data set *I-Front* hereafter. I-Front is the layer between the hot ionized gas and the cold neutral gas. Scientists are interested in the relationship between the turbulence of the I-Front and the formation of other chemical elements. For instance, one of the tasks in the contest is to verify whether the turbulent flows can enhance the formation of the element $H_2$ from the free electrons $H+$. In order to answer this question, the goal of our case study is to explore the relationship among the curl field, which is related to the turbulence, and the mass abundance of the elements $H_2$ and $H+$.

In the preprocessing stage, we downsampled the data set from its original resolution ($600 \times 248 \times 248$) by 4 to save the space, normalized the range of each time series from its original range to [0, 1] since we are interested in its relative change, and used K-mean clustering to group the time series for each variable to reveal the salient patterns and to specify the target trends. Via K-mean clustering, we found out that the time series in the curl and $H_2$ fields mainly contain peaks, while those in the $H+$ field looks like a step function. Therefore, we specified the target trends as an impulse function for both curl and $H_2$ fields and a monotonically increasing function for the $H+$ field. In Figure 12 (a) the target trends from top to bottom are for the curl field, $H+$, and $H_2$. The scale of the target trends was re-scaled to [0, 1] in order to detect the time step with strongest response to the target trends. We then set the error thresholds for the curl, $H+$ and $H_2$ to 0.3, 0.1 and 0.2 times their maximal distances, respectively.

From the clustered trend sequences, we found out that while the target trends for $H_2$ tends to lead $H+$, the target trends of the curl field can happen either later or earlier than the other two target trends, as shown in Figures 12 (b) and (c). By assigning green and red colors to the cases in Figures 12 (b) and (c), respectively, we can visualize the spatial distribution of both cases, as shown in Figure 12 (d). The volume rendered image provides a visual cue for the scientist to further compare the difference between the two regions.

It is noteworthy that several peaks are detected in the curl field in the top plot in Figure 12 (b), which are drawn as bounding boxes. This also demonstrates that SUBDTW can detect repeating patterns.

## 8 Limitations

In this section we discuss the limitations of our method and present possible future research directions.

First, the speed of SUBDTW is quadratic to the number of time steps and hence is the performance bottleneck in our system. Based on a GPU-based DTW implementation proposed in our previous work [15], which can speed up the DTW computation by as much as 40 times compared to using CPUs, a future work of ours will be to accelerate SUBDTW computation using GPUs.

Since DTW and SUBDTW are not scale-invariant, given a target trend, a sub time series with a similar pattern but at a different scale may not always be detectable except in the following two cases. Given two similar time series $f$ and $g$, the first case is when the ratio between each element of the two series $f[t]/g[t]$ is constant. The other case is when there exists a constant offset $f[t] - g[t]$ between them. While different heuristics can be applied, solution to this limitation will be data- or application-dependent. For instance, in our first case study,

since our goal was to understand the path of the hurricane eye, we browsed the data points along the path of the hurricane eye in order to decide the appropriate scale to specify the target trends for the pressure and wind magnitude. Another example is our second case study, where we were interested in the relative change. We normalized the range of all time series to $[0, 1]$, and specified the scale of the target trends to the same range.

Another limitation is the identification of important trend sequences. Currently, although the users can quickly identify the trend sequences that have common prefixes, they could be also interested in trend states not always from a common prefix. One future work to overcome this limitation is to support user queries represented as regular expressions, and have the system find all the sequences that satisfy the expressions. An example of using regular expressions for visualization is in [8].

Finally, constructing meaningful trend sequences depends on the distance thresholds, which are specified by the user. While our current system can quickly generate a new set of trend sequences since the clustering algorithm is efficient, the tuning of the thresholds can be still non-intuitive. In the future, we will consider other information such as the histogram of the distances of all identified trends to assist the specification of the thresholds. We will also investigate more accurate and robust filtering schemes to select salient data points. One direction is to use spatial coherence since the distances between neighboring data points should be small. For data points that have slightly larger distances than the thresholds, if their neighbors contain the target trends, these data points might contain the target trends as well and should be included.

## 9 Conclusion

We propose a novel method to visualize and explore multivariate time-varying data sets. Our main focus is to understand the temporal relationships among the variables. Based on a new distance metric SUB-DTW, we can estimate when and where a user-specified trend occurs in space and time. The extracted trends are modeled as trend sequences to highlight the correlations among different variables. Several visualization interfaces are proposed for the user to explore important trend sequences and visualize their spatial and temporal distributions. We tested our algorithm using scientific data sets and presented our findings to demonstrate the utility of our method.

## Acknowledgments

The authors would like to thank the anonymous reviewers for their comments. This work is supported by NSF ITR Grant ACI-0325934, NSF RI Grant CNS-0403342, NSF Career Award CCF-0346883, and DOE SciDAC grant DE-FC02-06ER25779.

## References

[1] H. Akiba, N. Fout, and K.-L. Ma. Simultaneous classification of time-varying volume data based on the time histogram. In *EuroVis '06: Proceedings of the Joint Eurographics - IEEE VGTC Symposium on Visualization 2006*, pages 1–8, 2006.

[2] H. Akiba and K.-L. Ma. A tri-space visualization interface for analyzing time-varying multivariate volume data. In *EuroVis '07: Proceedings of the Joint Eurographics - IEEE VGTC Symposium on Visualization 2007*, pages 115–122, 2007.

[3] C. L. Bajaj, V. Pascucci, G. Rabbiolo, and D. Schikorc. Hypervolume visualization: a challenge in simplicity. In *VVS '98: Proceedings of the IEEE Symposium on Volume Visualization 1998*, pages 95–102, 1998.

[4] J. Blaas, C. Botha, and F. Post. Extensions of parallel coordinates for interactive exploration of large multi-timepoint data sets. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1436–1451, 2008.

[5] J. A. Fails, A. Karlson, L. Shahamat, and B. Shneiderman. A visual interface for multivariate temporal data: Finding patterns of events across multiple histories. In *VAST'06: Proceedings of the IEEE Symposium On Visual Analytics Science And Technology 2006*, pages 167–174, 2006.

[6] Z. Fang, T. Möller, G. Hamarneh, and A. Celler. Visualization and exploration of time-varying medical image data sets. In *GI '07: Proceedings of Graphics Interface 2007*, pages 281–288, 2007.

[7] N. Fout, K.-L. Ma, and J. Ahrens. Time-varying, multivariate volume data reduction. In *SAC '05: Proceedings of the ACM Symposium on Applied Computing 2005*, pages 1224–1230, New York, NY, USA, 2005. ACM.

[8] M. Glatter, J. Huang, S. Ahern, J. Daniel, and A. Lu. Visualizing temporal patterns in large multivariate data using modified globbing. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1467–1474, 2008.

[9] L. J. Gosink, J. C. Anderson, W. E. Bethel, and K. I. Joy. Variable interactions in query-driven visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1400–1407, 2007.

[10] H. Guo, R. Renaut, K. Chen, and E. Reiman. Clustering huge data sets for parametric pet imaging. *BioSystems*, 71(1–2):81–92, 2003.

[11] S. Havre, E. Hetzler, P. Whitney, and L. Nowell. Themeriver: Visualizing thematic changes in large document collections. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):9–20, 2002.

[12] H. Hochheiser and B. Shneiderman. Dynamic query tools for time series data sets: timebox widgets for interactive exploration. *Information Visualization*, 3(1):1–18, 2004.

[13] A. Inselberg. Multidimensional detective. In *INFOVIS '97: Proceedings of the IEEE Symposium on Information Visualization '97*, pages 100–107, 1997.

[14] J. Johansson, P. Ljung, and M. Cooper. Depth cues and density in temporal parallel coordinates. In *EuroVis '07: Proceedings of the Joint Eurographics - IEEE VGTC Symposium on Visualization 2007*, pages 35–42, 2007.

[15] T.-Y. Lee and H.-W. Shen. Visualizing time-varying features with tac-based distance fields. In *PV'09: Proceeding of IEEE Pacific Visualization Symposium 2009*, pages 1–8, Los Alamitos, CA, USA, 2009. IEEE Computer Society.

[16] P. S. Mccormick, J. Inman, J. P. Ahrens, C. Hansen, and G. Roth. Scout: A hardware-accelerated system for quantitatively driven visualization and analysis. In *VIS '04: Proceedings of the IEEE Visualization 2004*, pages 171–178, 2004.

[17] K. McDonnell and K. Mueller. Illustrative parallel coordinates. *Computer Graphics Forum*, 27(3):1031–1038, 2008.

[18] P. J. Moran and C. Henze. Large field visualization with demand-driven calculation. In *VIS '99: Proceedings of the IEEE Visualization '99*, pages 27–33, 1999.

[19] P. Muigg, J. Kehrer, S. Oeltze, H. Piringer, H. Doleisch, B. Preim, and H. Hauser. A four-level focus+context approach to interactive visual analysis of temporal features in large scientific data. *Computer Graphics Forum*, 27(3):775–782, 2008.

[20] M. ten Caat and N. M. Maurits. Design and evaluation of tiled parallel coordinate visualization of multichannel eeg data. *IEEE Transactions on Visualization and Computer Graphics*, 13(1):70–79, 2007.

[21] C. Tominski, J. Abello, and H. Schumann. Axes-based visualizations with radial layouts. In *SAC '04: Proceedings of the ACM Symposium on Applied Computing 2004*, pages 1242–1247, 2004.

[22] C. Wang, H. Yu, and K.-L. Ma. Importance-driven time-varying data visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1547–1554, 2008.

[23] D. Whalen and M. L. Norman. Competition data set and description. 2008 IEEE Visualization Design Contest, 2008. http://vis.computer.org/VisWeek2008/vis/contests.html.

[24] J. Woodring and H.-W. Shen. Chronovolumes: A direct rendering technique for visualizing time-varying data. In *VG '03: Proceedings of the Third International Workshop on Volume Graphics 2003*, pages 27–34, 2003.

[25] J. Woodring and H.-W. Shen. Multiscale time activity data exploration via temporal clustering visualization spreadsheet. *IEEE Transactions on Visualization and Computer Graphics*, 15(1):123–137, 2009.

[26] J. Woodring, C. Wang, and H.-W. Shen. High dimensional direct rendering of time-varying volumetric data. In *VIS '03: Proceedings of the IEEE Visualization 2003*, pages 417–424, 2003.