WPI-CS-TR-10-06 March 2010

Visual Analysis of Multivariate Data Streams Based on DOI Functions

by

Zaixian Xie
Matthew Ward
and Elke Rundensteiner

Computer Science
Technical Report
Series

WORCESTER POLYTECHNIC INSTITUTE

Computer Science Department
100 Institute Road, Worcester, Massachusetts 01609-2280

Abstract

The analysis of data streams has become quite important in recent years, and is being studied intensively in fields such as database management and data mining. Although some researchers in data and information visualization have investigated the visual analytics of streaming data to a certain degree, there are some obvious limitations in existing work: (1) a lack of effective techniques to show how data patterns change over time; and (2) limited ability to represent multivariate correlations. In this paper, we propose a framework to visualize multivariate data streams via a combination of windowing and sampling strategies. In order to help users observe how data patterns change over time, we display not only the current sliding window but also abstractions of past data in which users are interested. Sampling is applied within each single sliding window to help reduce visual clutter as well as preserve data patterns. Further, we allow different windows to have different sampling ratios to reflect how interested the user is in the contents. We use a DOI (degree of interest) function to represent users' interest in the data within a set of windows. Users can apply two types of pre-defined DOI functions. An interactive tool also allows users to adjust the DOI function online, in a manner similar to transfer functions in volume visualization, to enable a trial-and-error exploration process. In order to visually convey the change of multidimensional correlations, we designed four layout strategies. User studies showed that three of these are effective techniques to achieve the above goal compared to traditional time-series data visualization techniques. Based on this evaluation experiment, we derived a guide to advise data analysts and visualization system developers on how to choose appropriate layout strategies in terms of the characteristics of datasets and data analysis tasks. Case studies are discussed to show the effectiveness of DOI functions and the various visualization techniques.

Keywords: Data stream, multivariate data, visual analysis.

1 Introduction

Advances in hardware enable people to record data at rapid rates, e.g., kilobytes or megabytes per second or even higher speeds. Some real application areas require data collection and analysis at such a high speed. Moreover, the newly acquired or generated data items often need to be processed immediately, as in many cases the volume of data precludes storage for later analysis. For example, network traffic monitoring involves tracking each packet to identify features of interest, such as bottlenecks and potential intrusions. In the areas of database and knowledge discovery, the term *data streams* or *streaming data* has been used to refer to such data that keeps growing and needs to be processed on the fly. Researchers have developed many techniques to manage, query and analyze data streams in real-time [6].

In recent years, people have agreed that visualization can play a critical role in the processes of data analysis and decision-making, since it can help analysts use visual perception to uncover different patterns, such as clusters, associations, relationships, and trends. Streaming data is similar to *time-series* data, which is identified as one of basic data types [18] in the area of information visualization. In both data types, each datapoint has a time attribute, i.e., a timestamp. One can find a rich set of visualization techniques for time-series data in the literature. If we directly apply existing time-series data visualization techniques to streaming data, we can partially address the problem of visually exploring data streams. For example, a continuously expanding line chart can convey the trend of a univariate data stream. However, an important characteristic of streaming data, namely *unbounded input*, makes this simple approach ineffective and incomplete, as existing visualization techniques for time-series data generally regard the whole dataset as static and assume that all of the data is available before rendering. This is impossible for streaming data. The designed techniques must be capable of processing data in a continuous and unbounded fashion.

Two other issues exist in the visual exploration of both time-series and streaming data that must be addressed in order to help users perform some common data analysis tasks:

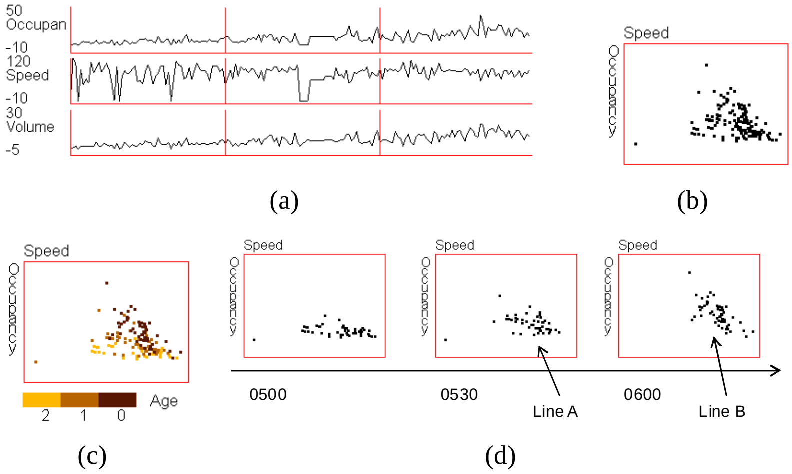
(1)**Temporal Visual Mining**: There exist many data mining tasks for time-series data, such as the discovery of temporal association rules and pattern evolution [17]. Existing time-series data visualization techniques only support a small fraction of these tasks. In this paper, we focus on two important types of temporal mining: data patterns for a specific time period, and how data patterns change over time.

(2) **Multivariate Correlations**: Although a few existing visualization techniques for time-series data try to present the relationships among multiple dimensions, their usefulness is often limited. For example, [9] shows the degree of importance for dimensions, and [8] presents some pre-specified statistical values among dimensions, but dimensions often have complex relations that these methods do not convey. In this paper, we aim to combine multivariate and time-series data visualization techniques to fill this gap.

The main goal of this paper is to present a framework for visually exploring unbounded multivariate data streams that can convey trends for each dimension, multivariate patterns, and the change in these patterns over time. To achieve this goal, one intuitive solution is to split the whole stream into non-overlapped sliding windows and send them through the visualization pipeline one by one, and provide an animation to users. This is certainly feasible, but actually is problematic. Because users can easily forget patterns shown to them in past frames, especially when the length of sliding windows is long, it is difficult for users to capture how data patterns change.

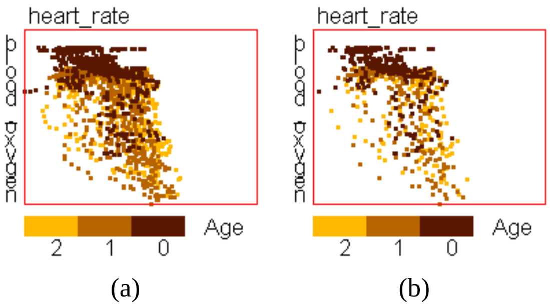
Our approach to achieving the main goal is as follows. We mix the data in the current window with those in the past ones in the same view and distinguish them via different visual attributes; or juxtapose these data in an ordered set of views. A DOI (degree of interest) function [5] is introduced to describe the degree of users' interest in a particular window. A lower DOI value results in a smaller sampling ratio. This approach works in two ways: (1) Users can choose which windows to show, normally those containing data patterns that users want to compare; and (2) Users can reduce visual clutter via assigning lower DOI values to the selected windows.

Figure 1 shows an example of our visualization layout. This figure uses a small slice (5:00AM-6:30AM on Feb. 16, 2009) of a traffic data stream provided by Mn/DOT (Minnesota Department of Transportation) [16]. In this slice, each datapoint includes three measured values during a 30 second period from sensor D722. We only choose two dimensions to investigate their correlations here. One dimension is the average vehicle speed (*Speed*), and the other is the percentage of time that the detector sensed a vehicle (*Occupancy*). We present a traditional time-series data visualization technique, line charts, in Figure 1(a). Figure 1(b) shows a naïve solution that treats all datapoints in these 1.5 hours as a static dataset. We can neither identify any strong relationship between *Speed* and *Occupancy*, nor learn how patterns change over time. Figure 1(c) splits the data stream into three sliding windows and uses colors to denote the age of the windows. We can draw a conclusion that *Occupancy* does not correlate with the change of *Speed* in the early period, but an obvious negative relationship exists between these two dimensions later. In Figure 1(d), each sliding window is visualized by a scatterplot and three of them are juxtaposed in terms of the time attribute. We can easily confirm the pattern we found via Figure 1(c), but the last visualization technique uses more canvas space than the previous ones. In Figure 2, we can see how DOI functions work to reduce visual clutter. After the DOI function is adjusted to reduce sampling ratios of three windows, visual clutter is reduced, and users can more clearly see how the main clusters move over time.



(a) (b) (c) (d)

Figure 1: Figures show some of the main ideas of this paper, using a traffic data collected from a highway entrance. (a) A traditional time-series data visualization; (b) All of datapoints are shown together via a traditional scatterplot; (c) The ages of data are denoted by colors; (d) Juxtaposition of data in the order of timestamps. Figures (c) and (d) can convey how data patterns change, but it is difficult for (a) and (b) to present this change.



(a) (b)

Figure 2: Using DOI functions to reduce visual clutter on a sleep data stream. (a) All datapoints are displayed; (b) Sampling is applied to each sliding window based on the DOI function after user adjustment.

The main contributions of this paper are as below:

- We present a framework for introducing windowing and sampling strategies into traditional multivariate and time-series data visualization techniques. Its aim is to handle unbounded input as well as conveying trends, multivariate correlations, and the changes of data patterns over time.

- This framework allows users to define DOI functions to describe the degree of users' interest [5] for different portions of the data. This function enables users to choose which windows to display, and adjust sampling ratios to reduce possible visual clutter.

- We provide four layout strategies to organize traditional multivariate data visualizations and convey the change of multi-dimensional correlations. User studies showed that three of these can effectively convey the multivariate pattern change compared to traditional time-series data visualization techniques. Using the experiment results, we also derived a guide to advise data analysts and visualization system developers to choose appropriate layout strategies in terms of the characteristics of datasets and data analysis tasks.

- We integrate interaction techniques to help explore data streams, including a DOI function interaction tool that helps analyze data via a trial-and-error process, and linked brushing across multiple views. Several cases studies are discussed to show the effectiveness of these interaction tools.

2 Related work

Streaming data visualization can be regarded as real-time time-series visualization with unbounded and large-scale input. In this section, we review existing visualization techniques for time-series data, focusing on the handling of large-scale input and representation of multivariate correlations. In addition, we also investigate some recent research achievements regarding visually exploring data streams.

Time-series Data Visualization: In order to deal with large time-series datasets, some abstraction algorithms have been introduced into time-series visualization for adapting large temporal datasets to limited display space. These can be categorized into two approaches: data-driven [15] and user-driven [10] means. Miksch et al. [15] developed an abstraction algorithm for temporal univariate data that aims to transform numerical values to qualitative descriptions. It can smooth data oscillation near thresholds. Hao et al. [10] used a sampling technique to abstract time-series data and introduced DOI (degree of interest) functions to determine the sampling rate. The DOI function is used to represent how users are interested in different portions of a time-series dataset. The subset of the original dataset with a higher DOI value is abstracted using a higher sampling rate and displayed in higher resolution. Otherwise, an overview with lower resolution will be displayed. Hao's DOI function is designed for only static time-series data, and does not consider periodic phenomena. We borrowed this idea and adapted it to streaming data. We also describe two types of DOI functions, one of which can help users explore data streams having repeated patterns with a certain cycle. We also got inspiration from other works on time-series data because of the similarities between temporal and streaming data.

Data Stream Visualization: Some visualization techniques and systems have been designed and implemented for particular types of data streams. Some researchers focus on univariate data. Hao et al. [11] used variable resolution density displays to visualize univariate data. They designed circular overlay displays to avoid data shift movements after the display is full, thus avoiding the difficulties for users to observe visualizations with too much change between frames. *BinX* [3] is a real-time system to visualize time-series data on the fly. It uses an aggregation algorithm to adapt large datasets to a limited canvas and supports online adjustment for the levels of aggregation. Our work will focus on how to convey multidimensional data patterns in data streams, which is significantly different from these existing efforts.

Several recent research efforts involve the visualization of multidimensional correlations. Wong et al. [22] present techniques for handling a multidimensional data stream, similar to our work. However, their focus is how to reduce the time complexity for scaling algorithms to generate scatterplots for visually conveying clusters in data streams. Their basic approaches include data stratification that intelligently reduces the data size using wavelets or sampling, and data-fusion to project new data items onto the existing visualization to avoid re-processing the whole dataset. Thus the problem they solved is to visualize the whole data stream in one pass, which is different from our goal to convey the data patterns within a window and the change of data patterns over time. Yu et al. [24] developed a tool for the visual analysis of a multimedia multi-stream data. Some continuous time-series data and event data are first extracted from the multimedia stream, and then are visualized via line charts, gray-level bars and color bars. Users can highlight selected data portions, or zoom in on the region of interest to study the data trends and the multivariate data patterns. Compared to the techniques we present in this paper, Yu's tool focuses only on some fairly simple patterns. In addition, they did not consider the unbounded nature of data streams.

Some existing research involves text streams. For instance, *TextPool* [1] is a tool for visualizing and maintaining an up-to-the-minute understanding of live streams of text such as newswires and closed-captioned television. News stories are represented by content vectors, which are calculated from news' titles and a 10-30 word description. The final visualization is an animation of a graph in which nodes represent salient terms from the streams. Compared to our work, *TextPool* does not consider how to represent the change of patterns in a way other than animation, in which users can easily forget the information in prior frames.

3 Streaming Data Model

To formalize the modeling of arriving data elements, we use the following definition derived from [2]:

$$(V,ts) = (v_1, v_2, ..., v_n, ts) \tag{1}$$

to describe one arriving data element, which we call a *datapoint* in the remainder of this paper. Note that n is the number of dimensions, $v_i (1 \leq i \leq n)$ are real numbers, and ts is the timestamp that represents when the datapoint originated. In this paper, we do not consider nominal values or other types of data streams, such as documents, images and video. We consider these types as part of our future work.

There are different types of streaming datasets in terms of their semantics. In this paper, we focus on a widely used type, namely *Univariate-Aggregation*, in which each dimension can be regarded as a univariate data stream, e.g., the traffic data stream we mentioned in Section 1. There also exists other types of data streams. For instance, arriving datapoints may belong to different objects, so trends on each dimension do not always make sense. The type of data stream has a significant impact on the choice of visualization strategies. For the univariate-aggregation type, our main goal is to convey both trends for each dimension and multivariate data patterns. In the future, we plan to explore other stream types.

The two streaming datasets used in this paper are the following.

Traffic Data Stream: In Section 1, We showed a slice of this data stream, which is provided by Mn/DOT [16]. Mn/DOT installed more than one thousand sensors on highway entrance/exit ramps and main lanes throughout the Twin Cities Metro area. Each detector can collect a value for each of the following measures with an interval of 30 seconds: (1) Volume: the number of vehicles passing the detector. (2) Occupancy: the percentage of time that the detector sensed a vehicle. (3) Speed: the average speed of all vehicles passing the detector. The website of Mn/DOT provides a Java-based tool, *DataExtract*, to allow users to extract detector data to csv files. Thus we can get several thousand values every 30 seconds. Instead of using all of these values at the same time, we select one detector and retrieve its three measures during a specific time period, e.g., one day or week.

Sleep Data Stream: This data stream is a physiological dataset (Santa Fe time series competition data set B) selected from the PhysioBank archive [7]. It is recorded from a patient suffering from sleep apnea in a sleep laboratory. Since it is relatively long (about 4 hours at a frequency of 2Hz), we use it to simulate a data stream. This dataset has three measures: heart rate, chest volume (respiration force), and blood oxygen concentration.

4 The Framework Based on Windowing, Sampling and DOI Functions

Before discussing the framework in detail, we introduce two terms:

The **Sampling Ratio** is the percentage of datapoints to be selected to display. Although some researchers use the term *sampling rate*, it is easy to confuse readers because sampling rate normally refers to the number of samples per time unit taken from a continuous signal [13]. The definition of sampling ratio used in this paper is as follows:

$$r(\text{sampling ratio}) = \frac{\text{the number of selected datapoints}}{\text{the number of all datapoints}} \tag{2}$$

Note that sampling ratio r must satisfy $0 \leq r \leq 1$.

DOI Functions represent how interested the user is in seeing a particular sliding window. In a regular static dataset, a DOI function calculates a value to represent the degree of interest for a portion of the dataset. Then the portion of data with high DOI values will be displayed with more detail [5]. For data streams, the story becomes a little complex. When the stream system gets a new sliding window (Window 1), a specific DOI level should be applied to this new portion of data. However, when Window 1 expires and a new window (Window 2) becomes the current one, users might want to focus on Window 2 and show less interest in Window 1. In this situation, the sampling ratio for Window 1 must change. Hence, the DOI function should have two parameters, a timestamp representing the specific sliding window and the current time point. Formally, the DOI function is given as DOI $= f_{doi}(t_d, t_c)$, where t_d and t_c are the timestamps corresponding to a specific sliding window and the current one. We use the smallest timestamp for all datapoints in the window as t_d and t_c. Other options are possible, e.g., the average timestamp.

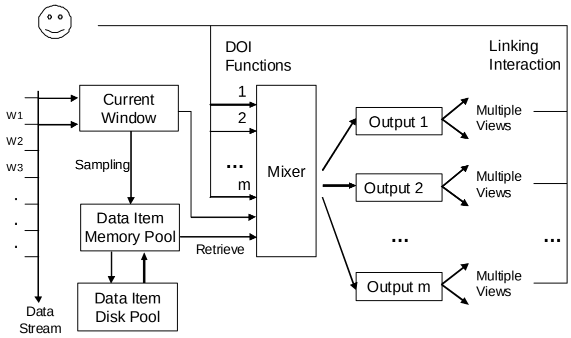


Figure 3: The framework of user-driven multiple-view visualization for data streams.

Figure 3 shows the proposed framework. Here we use non-overlapped sliding windows. In this figure, *data item memory pool* and *data item disk pool* reside in a piece of memory and in secondary storage, respectively. When a sliding window is formed, the data will be transferred to the memory pool. Because the data stream is infinite in nature, a part of the infrequently accessed data in the memory pool will be moved to the disk pool via data compression when the memory is full. The data staying in the disk pool also will be brought back the memory pool when users need to view some older data, though some will be lost during compression. The *Mixer* is the core of the whole framework. Its input includes the datapoints in the current window and a part of the older windows from the memory pool. It can assign sampling ratios and generate output that mixes datapoints from these windows. The sampling ratio for a given sliding window is calculated by a function $r = f_r(\text{DOI})$, where DOI is the output of the DOI function. Since we allow users to provide DOI functions as input, we call this framework *user-driven*.

We allow users to define multiple DOI functions to get more than one output because different tasks need different DOI functions. For example, during traffic monitoring, users might have two data analysis tasks: (1) identifying how vehicle speed changes within the recent hour; (2) comparing the traffic of today with that of yesterday at the same time. Obviously, two different DOI functions are necessary.

For each output, our framework can provide multiple views to users. The possible relationships among views for the same output include: (1) They utilize different visualization techniques to convey different data patterns; (2) Each view visualizes a part of the data, e.g., the recent *k* windows are juxtaposed into *k* views to form small multiples [20]. Thus a user can watch how data patterns change over time via comparing multiple views. We also allow users to interact with multiple views using linking operations. One example of linking is that users can choose one region of interest in a line chart, and then datapoints falling into this region will be highlighted in a scatterplot matrix. We will show ways to organize multiple views in Section 6 and describe more about interaction techniques in Section 7.

As discussed before, we need to switch data between the memory pool and the disk pool. Since data streams are by nature infinite, the disk pool will eventually be full. One solution is to use lossy data compression techniques that lose some data detail but keep the primary data patterns. For older data, we can allow the loss of more detail than the more recent data. In this paper, we focus on the visualization and interaction techniques. The storage issues will be described via a future paper.

5 DOI Functions

In this section, we describe two types of DOI functions that can be used for some common tasks. As discussed in Section 4, the output of a DOI function $f_{doi}(t_d, t_c)$ is a DOI value, which needs to be mapped to a sampling ratio via the function $r = f_r(\text{DOI})$. For the sake of convenience, we define DOI functions in a way that their output is just the sampling ratio.

Type RC (Recent Change) : Figure 4(a) shows the curve for this type of DOI function. It aims to help users study how data patterns change within the recent $k+1$ sliding windows, which are assigned sampling ratios r_0, r_1, ..., r_k in the order from the current window to the past ones. Note that we do not require that $r_i \neq r_j$ when $i \neq j$. One common usage is to let $r_0 = r_1 = ... = r_k = 1.0$. Figure 1(c) is generated using this type of DOI function with arguments $k = 2$ and $r_0 = r_1 = r_2 = 1.0$. If there is too much visual clutter and users are less interested in the old data, we can let $r_i < 1$ for $1 \leq i \leq k$.

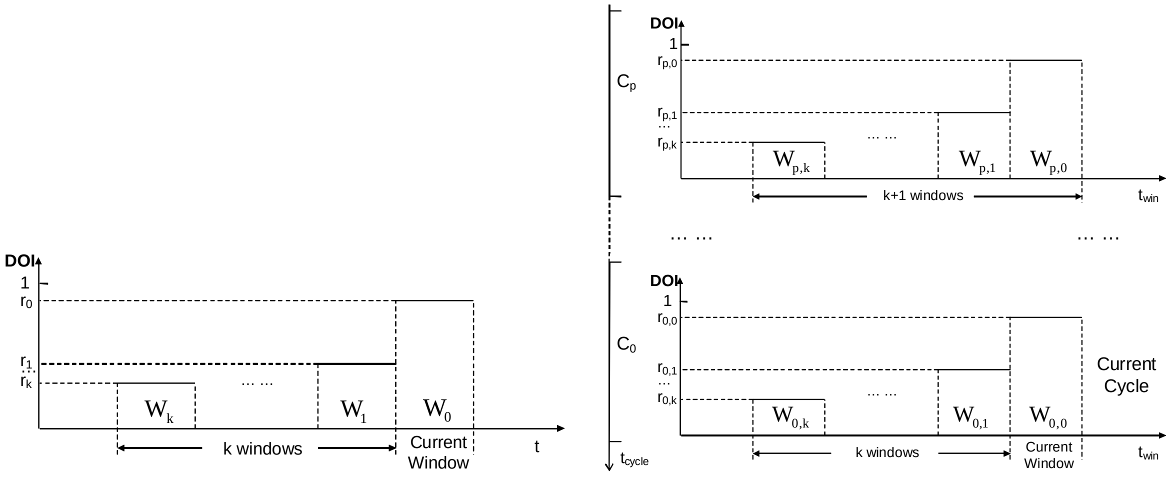
Type PP (Periodic Phenomena) : The DOI functions shown in Figure 4(b) can assist users in observing data patterns with periodic characteristics. The data stream is split into multiple cycles (the vertical time axis) with the same length. Each cycle contains multiple sliding windows (the horizontal time axes). In each cycle, the DOI function has a shape similar to Type RC functions. The DOI function in Figure 4(b) enables users to investigate data patterns within the recent $p+1$ cycles. $W_{0,0}$ is the current window, and $W_{i,0}(1 \leq i \leq p)$ belong to the past cycles, but have the same position in the cycle as $W_{0,0}$. In each cycle, this function also chooses k windows just before $W_{i,0}(0 \leq i \leq p)$ to display. Thus it can help users study how data patterns change across both windows and cycles. Consider the example of monitoring traffic. Imagine the current sliding window is 6:00AM-6:30AM on a Monday. The current traffic data pattern could be similar to last Monday, and less similar to last Tuesday to Friday, and totally different from last weekend for the same interval (6:00AM-6:30AM). To confirm this assumption, we can define a Type PP DOI function to choose only sliding windows corresponding to 6:00AM-6:30AM in these days.

The DOI function we defined is similar to the opacity transfer function in volume rendering [12]. The opacity transfer function assigns an opacity value to a voxel based on voxel's intensity and can bring out certain feature of those voxels having high opacity values. The relationship between the sampling ratio and the sliding window timestamp is like the relationship between the opacity value and the voxel's intensity.

6 Visualization Techniques

As we mentioned in the Introduction, our goal is to visually convey the change of multidimensional correlations. Thus we designed the visualization techniques with the following question as the main consideration: How do we organize datapoints in different sliding windows to convey multivariate correlations and the changes of data patterns? Obviously, it does not work to directly visualize the Mixer output via a traditional multivariate visualization technique, such as parallel coordinates and scatterplot matrices (see Figure 1 (b)). Such a solution blends data patterns of all windows chosen by the DOI function in the final visualization. It is almost impossible for users to retrieve data patterns for a particular time period and investigate how patterns change over time.

In this section, we will first introduce four layout strategies, namely *superimposition, juxtaposition*, *step juxtaposition* and *animation playback*, to answer the above question, and demonstrate



(a) (b)

Figure 4: Two instances of DOI functions: (a) Type RC (Recent Change); (b) Type PP (Periodic Phenomena).

their usage with type RC DOI functions. Note that these strategies can be applied to any multivariate visualization techniques; we mainly use scatterplots as examples. These four strategies will then be extended to type PP DOI functions. Finally, we develop a new visualization technique, namely "embedded view", via combining line charts and scatterplot matrices. This is to take advantage of the visual representation capabilities of multivariate and time-series data visualization techniques in one figure.

6.1 Layout Strategies

Superimposition: This strategy puts all datapoints in a single picture, but distinguishes datapoints from different sliding windows via visual attributes, the choice of which, obviously, can affect the effectiveness of final visualizations. Xie et al. performed a user study on visual representation of data quality and concluded that hue has a stable capacity to convey data attributes under parallel coordinates and scatterplot matrices as long as the visualization is not too cluttered [23]. The reason is probably that it is processed preattentively [21] and does not require extra space. Thus we decided to use colors to convey the timestamps of sliding windows. Figure 1 (c) is generated via applying superimposition to a scatterplot.

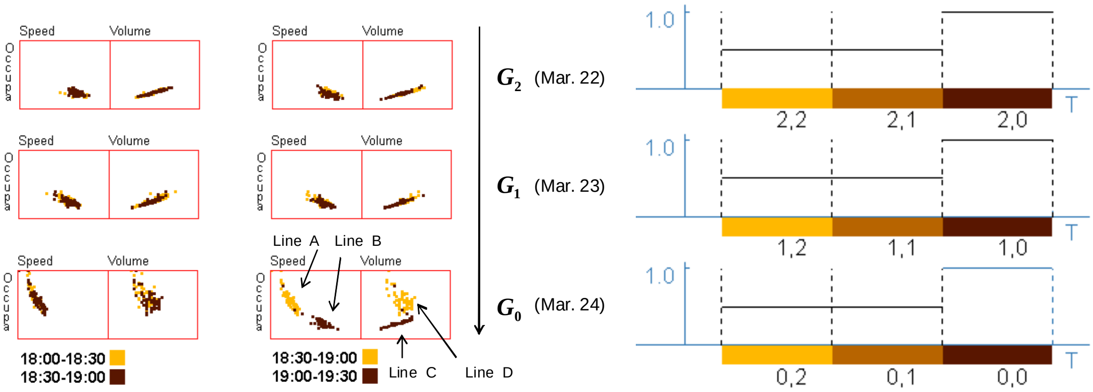
An obvious disadvantage is that displays can become overloaded with too much information, which may result in a longer analysis time. Moreover, if there are too many windows to be chosen in the DOI function and many of the datapoints from different sliding windows overlap each other, it is difficult to distinguish them, even if we use color to convey the window to which they belong.

Juxtaposition: In this method, we generate one *sub-picture* using a multivariate visualization for one time window, and then place these figures in order of time (horizontally, vertically, or a grid). In Figure 1(d), each scatterplot holds the datapoints from one sliding window. Users can see the change of data patterns via comparing three sub-pictures.

Although juxtaposition can overcome some shortcomings of superimposition, it brings two new disadvantages: (1) Let us recall Figure 1(d), in which the dots in the second and third sub-pictures formed two lines, A and B. As a recognizable difference exists between the slopes of lines A and B, users can draw conclusions about the change of data patterns. If this difference is not that big, users may not easily identify the change of line slopes using juxtaposition, as there is some distances between these two lines. In the superimposition layout, this difference should be recognized more easily than juxtaposition, assuming there is not too much visual clutter, because one line can be regarded a reference when users observe the other. Thus superimposition has a stronger capability to help users identify subtle changes of patterns than juxtaposition. (2) If users want to compare the data patterns between two windows, they must move their eyes back and forth. This could make the data analysis tasks cumbersome and might result in a longer response time, especially when there are a large number of windows in the DOI functions.

In order to overcome the shortcomings from both superimposition and juxtaposition, we developed a third layout strategy to combine the advantages of the above two strategies, namely *step juxtaposition*.

Step Juxtaposition: Imagine the DOI function chooses $k+1$ (See Figure 4(a)) windows to display. We create k sub-pictures: the first shows W_k and W_{k-1}, the second presents W_{k-1} and W_{k-2}, and so on. This strategy uses superimposition to help users compare the data patterns of two contiguous windows, juxtaposition to reduce possible visual clutter and shortens completion time for data analysis tasks. Figure 5 shows an example. More than 2 windows can be superimposed in one sub-picture in this technique to save canvas size if no too much visual clutter.



(a) (b)

Figure 5: A step juxtaposition output using a type PP DOI function with the grouping approach GA1, which is shown in Figure (b). The cycle length is one day. In Figure (a), We can see clearly how data patterns change within the recent three sliding windows for March 24. However, data patterns do not have significant changes on March 22 and 23.

A more convincing example is shown in Figures 6 and 7, where we use a slice of traffic data (Sensor D722, Feb. 16, 2009). The DOI function is of type RC and 25 windows are selected. Imagine we were asked to find when the fit line slope changes from one window to the next. This is definitely impossible via superimposition since human eyes cannot effectively distinguish 25 colors in one figure. In figure 6, it is an arduous task because of too many windows. However, in Figure 7, this task becomes much easier. In each scatterplot, we only need to use light yellow datapoints as the reference and observe dark yellow dots. We can not only find obvious changes from Window 05:00-05:30 to Window 05:30-06:00, and from Window 05:30-06:00 to Window 06:00-06:30, but can also perceive tiny changes from Window 06:00-06:30 to Window 06:30-07:00, from Window 09:00-09:30 to Window 09:30-10:00, and from Window 09:30-10:00 to Window 10:00-10:30. These tiny changes are almost impossible to detect using juxtaposition (Figure 6). In the section on our user studies, we will see that step juxtaposition can help users obtain a much higher response accuracy than juxtaposition and shorten completion time for data analysis tasks.

Animation: It is an intuitive idea to play the data pattern change using an animation, with each frame representing a time window. Animation combines the benefits of the prior three visualization techniques:

(1) Because of the short memory of the human visual system, users can normally memorize the previous frame in the animation when the current frame is shown to us. Thus it has similar capabilities to convey data pattern change as superimposition and step juxtaposition.

(2) Compared to superimposition and step juxtaposition, animation can avoid the visual clutter caused by overlapping datapoints from different time windows.

(3) Unlike juxtaposition and step juxtaposition, animation still uses a canvas having the same size as superimposition, which can also avoid the possible visual clutter caused by overlapping resulting from a smaller canvas size.

However, animation can only highlight the change between a small number of contiguous time windows. Another shortcoming is that it might delay the data analysis tasks, because users frequently need to play the animation multiple times to confirm what they found. Moreover, we must show a window ID together with the visualization, so users know that window they are viewing. Thus users have to observe this caption while watching the animation and cannot fully focus on the data patterns.

Based on the above description and analysis of our proposed layout strategies, we list and compare their characteristics in Table 1. In Section 8, we will describe an experiment to compare the representation capabilities of these four layout strategies, and then derive a guide to advise analysts on choosing appropriate techniques for their data analysis requirements.

	Superimposition	Juxtaposition	Step Juxtaposition	Animation
(1) Contiguous Capabilities	Good	Fair	Good	Good
(2) Non-contiguous Capabilities	Good	Fair	Fair	Bad
(3) Overlapping	Much	No	Few	No
(4) Canvas size	Full	Shrunk	Shrunk	Full

Table 1: The comparison among four visualization techniques. The first column is the abbreviation for what we want to compare: (1) Contiguous capabilities: To what extent can this strategy convey the data pattern change between two contiguous sliding windows; (2) Non-contiguous capabilities: This is similar to the first aspect but focuses on the pattern change between two non-contiguous sliding windows; (3) Overlapping: The amount of possible visual clutter caused by overlapped data from different sliding windows; (4) Canvas size: The size of the basic visual unit (a scatterplot in our example) from which users can retrieve data patterns.

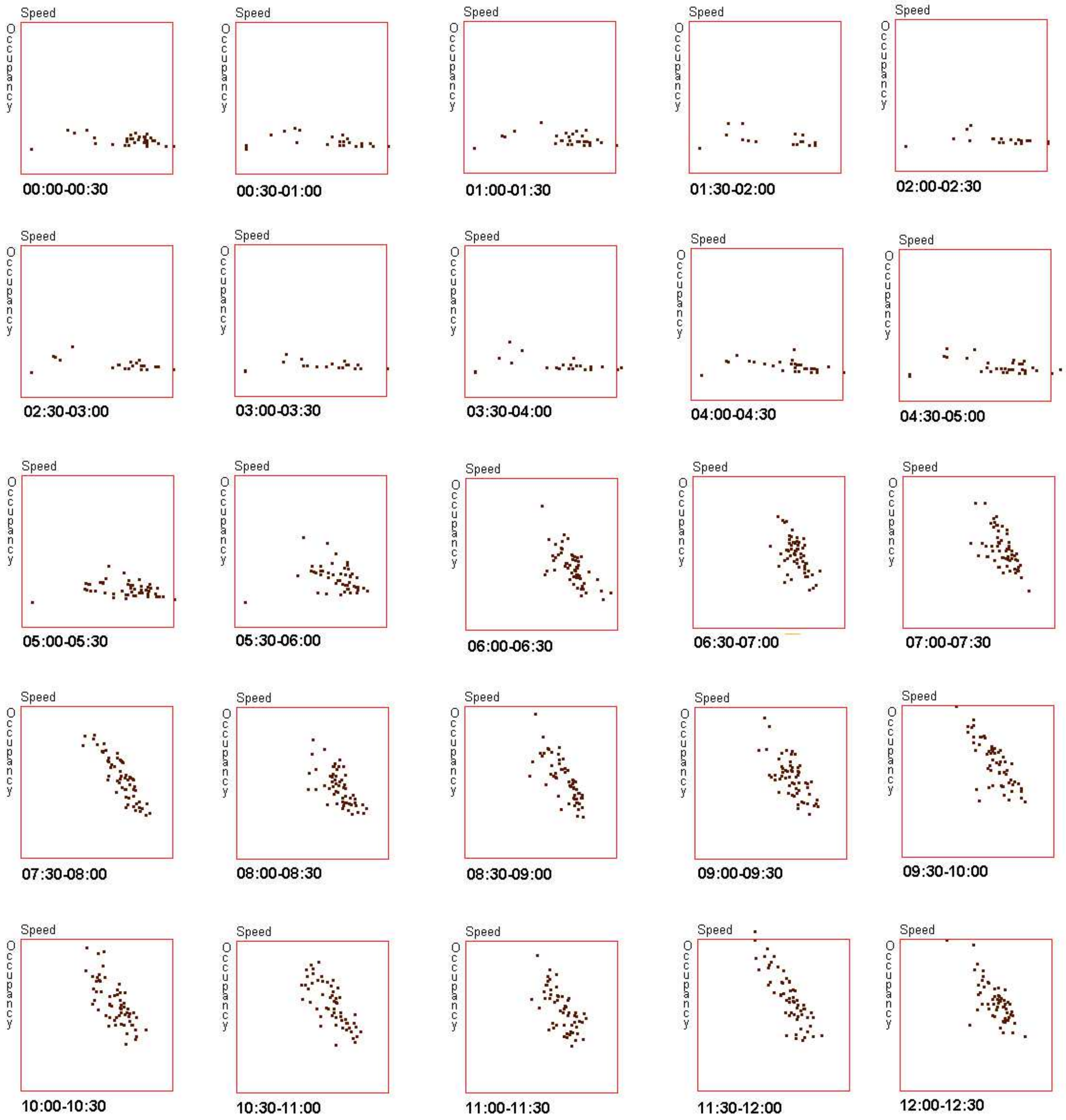


Figure 6: A juxtaposition output using the traffic data from sensor D722 on Feb. 16, 2009. Assume that the data analysis task is to detect the slope changes for fit lines of linear trends between contiguous windows. It is difficult to detect tiny slope changes. Moreover, even for an obvious change, it takes a longer time.

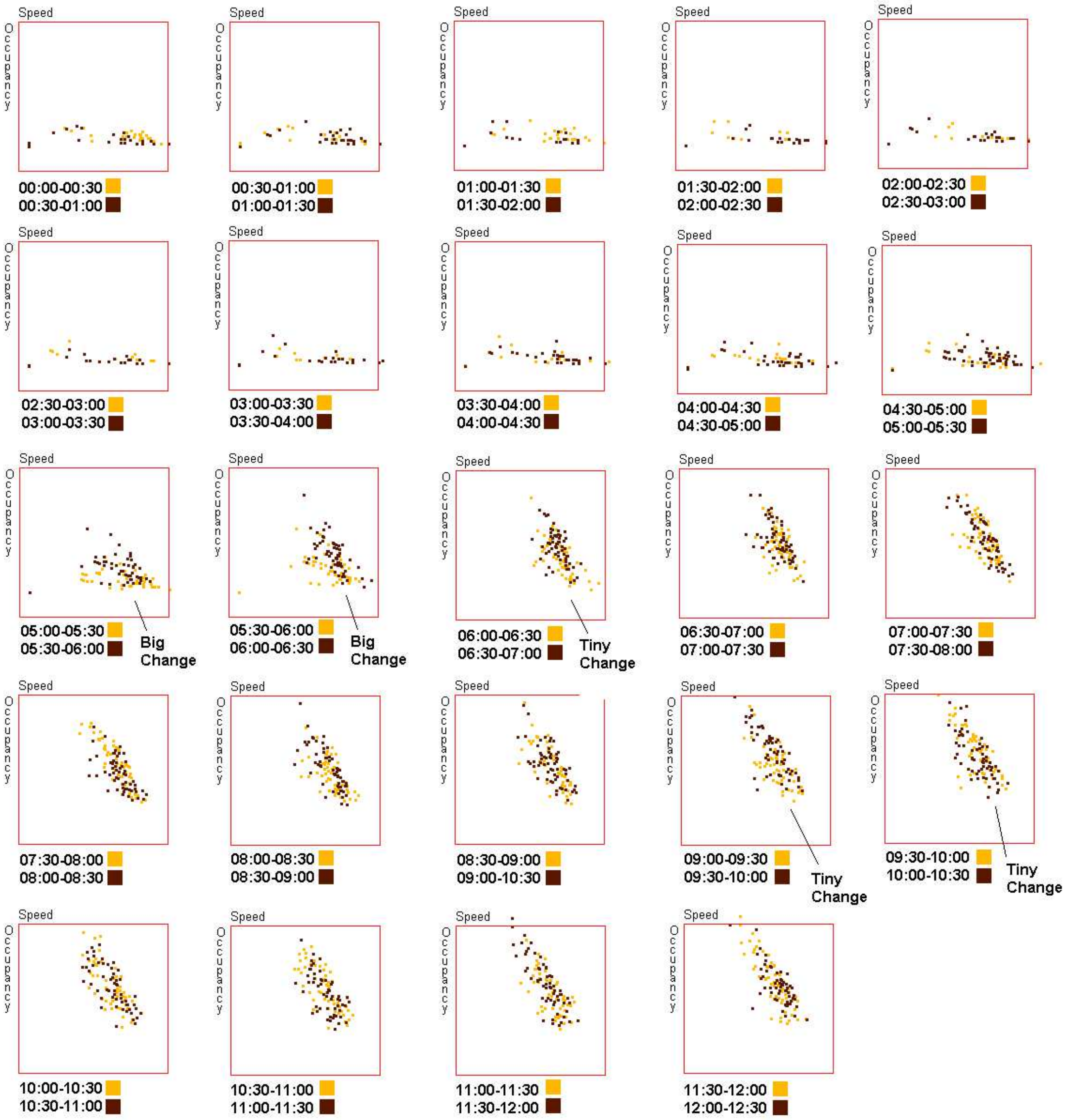


Figure 7: A step juxtaposition output using the same data as Figure 6. We can easily and quickly find when the slope of fit line for linear trend changes.

6.2 Extension of Layout Strategies to Type PP DOI Functions

We have discussed four layout strategies and showed their usage together with type RC DOI functions. In theory, we can directly apply these layout approaches to the case with type PP DOI functions. For example, in a type PP DOI function, users might choose two cycles and three time windows per cycle, and thus totally six windows should be shown. Such an example associated with traffic data is shown in Table 2. If we directly utilize the proposed layout strategies to handle these six windows, users could retrieve the information they want. However, this initial idea is not efficient compared to an alternative approach to grouping and then visualizing. This approach is based on a fact that users normally have two types of interests: (1) the pattern change across windows in the same cycle; and (2) the change of patterns across cycles in the same time period, such as window 1 and window 4. If users are interested in (1), we can organize two groups: (a) windows 1 & 2 & 3; and (b) windows 4 & 5 & 6. For the second task, we can split all windows into three groups: (a) windows 1 & 4; (b) windows 2 & 5; and (c) windows 3 & 6. The rationale is to put those windows in which users want to detect pattern changes into the same group. Then, we can use the four proposed layout strategies to visualize each group, respectively. Therefore, users can observe each group and try to extract the information of interest. Obviously, this grouping approach makes the pattern change analysis easier than the initial non-grouping method. For superimposition, it can decrease the number of time windows in one figure; and in the other three layout strategies, the grouping approach will put together only those windows in which users want to detect the pattern changes.

	6AM-7AM	7AM-8AM	8AM-9AM
Yesterday	Window 1	Window 2	Window 3
Today	Window 4	Window 5	Window 6

Table 2: Six windows used to explain the layout design for a type PP DOI function.

To be general, we provide two grouping approaches, called GA1 and GA2 (Figure 8) for the type PP function shown in Figure 4(b).

GA1: If the data analysis task focuses on the pattern change across windows within one cycle, $p+1$ groups (G_0, G_1, ..., G_p in Figure 8) will be provided. Actually, each group contains all windows in one cycle. An example of this grouping approach is shown in Figure 5.

GA2: If users are interested in changes across cycles, we generate $k+1$ groups (G'_0, G'_1, ..., G'_k in Figure 8). Every group has $p+1$ windows, each of which belongs to a cycle. All are in the same position within the cycle.

6.3 Integrating Time-series and Multivariate Data Visualizations

All of the above techniques assume the use of multivariate visualizations to convey multi-dimensional correlations. Another normal data analysis requirement in exploring data streams is to observe the trends for each dimension, which can be achieved using traditional time-series data visualization techniques such as line charts and heat maps. It is true that we can just put line charts and a scatterplot matrix side by side to convey both the trends for each dimension and multi-dimensional

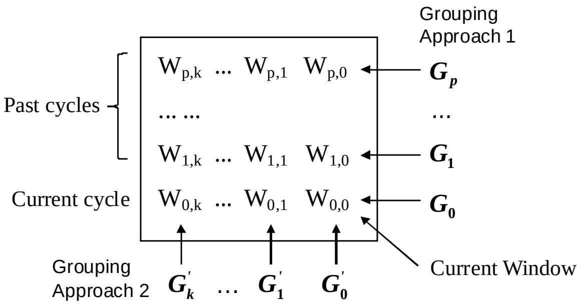
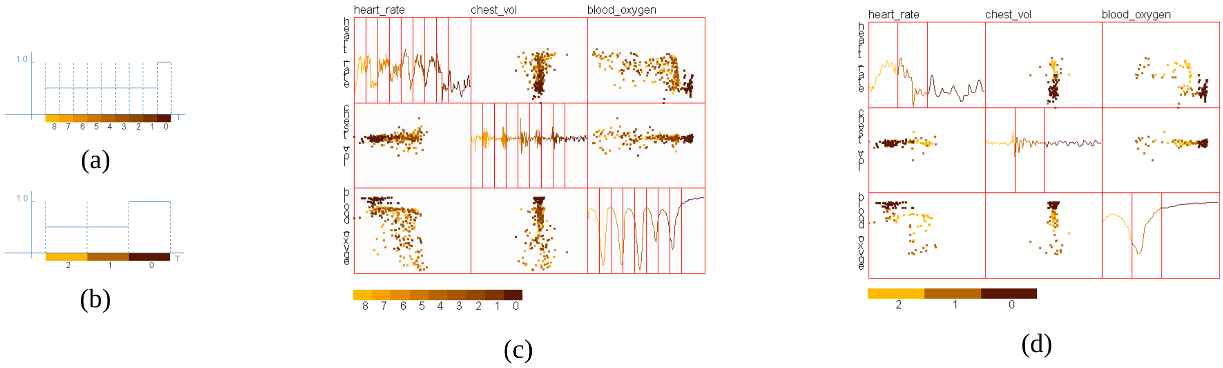


Figure 8: Two grouping approaches to helping users achieve various data analysis goals.



(a)

(b)

(c)

(d)

Figure 9: The embedded views for the sleep data stream. We not only see how clusters move over time in the scatterplots, but also see the trends for each dimension via line charts in the diagonal plots. Figure (c) is generated using the DOI function shown in Figure (a), which chooses the recent 9 windows to display. After the user uses the DOI function interactive tool to adjust the function to Figure (b), we can get a new view shown in Figure (d). Users can more clearly see the move of clusters on Figure (d) than Figure (c).

correlations. This requires decreasing the canvas size for each visualization since the total canvas size is normally fixed, e.g., monitor size. To overcome this shortcoming, we propose a novel technique, namely *embedded views*, to embed line charts into scatterplot matrices to save canvas space. This approach is adapted from the enhance scatterplot matrices of Cui et al. [4], who introduced 0D, 1D and 2D visualizations, including histograms, line charts and images, into the diagonal plots.

Figures 9(c) and 9(d) show two embedded views using the sleep data stream. They use the DOI functions shown in Figures 9(a) and 9(b), respectively. From these two views, we can clearly see how the main clusters move over time in the scatterplot. In addition, we can also study the trends for each dimension via line charts.

7 Interactions

7.1 DOI Function Interactive Tool

Although we described two pre-defined types of DOI functions in Section 5, it is necessary to enable users to define DOI functions by themselves to analyze data streams in different applications. We also feel that it will make the system much more useful to allow users to adjust the DOI functions interactively. Basically, visual analysis based on the DOI function is often a trial and error process. It is normal that analysts do not know the exact characteristics of the data patterns and how these patterns change prior to exploring the data streams. By allowing users to adjust the DOI functions, analysts can select a predefined DOI function first, and then adjust it to find useful data patterns while the system is running. Some possible adjustments to facilitate exploration include: (1) Increasing the sampling ratio to see more details or to decrease the ratio to avoid visual clutter. (2) Changing some of the arguments for pre-defined types of DOI functions. For example, if the number of sliding windows to be displayed for the type RC DOI function is large, say 9, but we find most important changes occur within the recent two or three windows, we can reduce it and observe the change in more detail.

We designed an interface to enable users to change the DOI function interactively. Using this tool, users can (1) drag the DOI function curve to change DOI values for a particular window; (2) save or load a DOI function to/from a file; (3) add/delete a window;(4) add/delete a cycle (only applicable for type PP function); and (5) reset the DOI function to the original state. Figures 2 and 9 show the effect of using this tool. In Figure 2(a), the arguments for the DOI function are set as $r_0 = r_1 = r_2 = 1.0$, but they are changed to $r_0 = 0.5$ and $r_1 = r_2 = 0.33$ in Figure 2(b). In Figure 9 we show the effect of reducing the number of windows.

7.2 Linked Brushing among Multiple Views

In order to help users explore subsets of interest, we introduce linked brushing into the framework. Brushing is a commonly used interaction technique to allow users to select a subset of data via a query. Linked brushing is used in multiple view visualizations. Users can link multiple views for one dataset. For instance, brushing points in one view can cause the same points to be highlighted in other views.

The linked brushing in our framework is similar to others. When users choose a subset in a view, the data in this subset will also be highlighted in other views. Two technical issues warrant discussion: (1) how to define a query for the subset of interest; and (2) how to highlight this subset. The second one is challenging. Many visualization tools, such as XmdvTool [14] and GGobi [19], use colors to highlight subsets of interest. However, in our framework, colors have been assigned to denote data age.

Query Definition: We use a hyperbox proposed by Martin and Ward [14] to specify a brush query, but add the time attribute to the brush definition, namely an *N+1-dimensional brush*:

$$([S_1,E_1],[S_2,E_2],...,[S_n,E_n],[S_t,E_t]) \tag{3}$$

where $[S_i,E_i](1 \leq i \leq n)$ denotes the start and end values for brush coverage on dimension *i*, which specify a range in which users are interested, and (S_t,E_t) denotes the time range. For a datapoint $(V,ts) = (v_1,v_2,...,v_n,ts)$ in the data stream, if it satisfies $S_i \leq v_i \leq E_i$ for all $i(1 \leq i \leq n)$ and $S_t \leq ts \leq E_t$, then it falls into the subset to highlight.

We allow users to use *data driven brushing* [14] to specify the query, which requires that the data falls into specified ranges on only one or two dimensions, and can be any values on other dimensions. The final visualization will show the exact bounds for those data of interest on other dimensions, including the time attribute.

Highlighting:

We solve the highlighting issue via the following two rules:

Rule 1: If a visual attribute other than color is available and is effective in denoting window age, switch to this new one and use colors to highlight selected datapoints; otherwise, go to rule 2. For example, if we select star glyphs as the multivariate visualization, we can place glyphs in the order of time, and use colored grids to denote the age of sliding windows.

Rule 2: We apply a fog effect to the data portion in which users are not interested by blending them with a gray shade. This results in these data being shown in a dimmer color, but users still can identify the ages of these data in terms of the color key. Figure 10 shows the fog technique for highlighting data, and data driven brushing to specify the query.

8 Evaluations on Layout Strategies

Earlier we proposed four layout strategies to utilize traditional multivariate visualizations to convey the pattern change in data streams. We also compared these four strategies using four criteria, and demonstrated some examples to show that our proposed techniques can effectively visualize the change of multivariate correlations. However, the answers to two important questions are still pending:

- Are the proposed techniques significantly better than other traditional time-series data visualization techniques, such as line charts and heatmaps, to convey the change of multidimensional correlations?

- Which layout strategy is the best to convey the pattern change for a specific dataset and pattern?

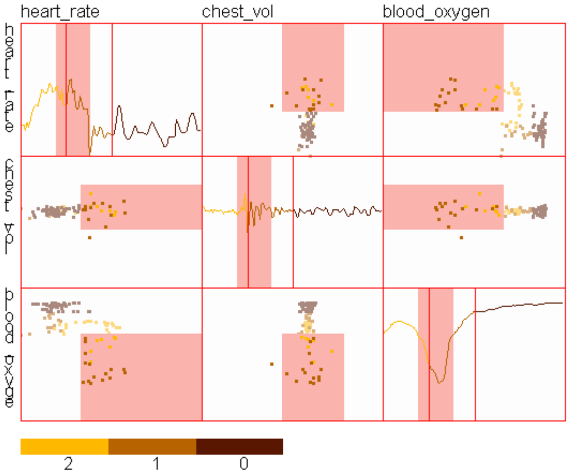


Figure 10: Data driven brushing is applied to the sleep data stream. Users first set the ranges for *Heart_rate* and *blood_oxygen*. The exact bounds of selected data on *Chest_vol* and timestamp are shown using a shade background.

For the first question, we might say yes. The reason is as follows: multivariate visualizations, such as scatterplot matrices, are specially designed for conveying multi-dimensional correlations, and the rationale of our techniques is to layout multiple multivariate visualizations, and show the pattern change. But line charts and heatmaps are designed for conveying trends of one dimension. Thus they cannot show multi-dimensional correlations or their changes very well. However, we can find some techniques in the literature that convey multi-dimensional correlation via line charts and heatmaps [24]. Thus a user study will likely be more convincing than the above argument.

The answer to the second question can help us design an efficient streaming data visualization system. If we do not have this answer, one possible solution is to provide all possible views to users and allow them to choose which one to use. When the system is running, users have to switch between various views to look for the best one. This could make users miss important pattern changes. The reasons include: (1) one kind of change might be obvious in one technique, but not evident in others; and (2) data patterns are not always changing from one window to the next, so users have no idea whether they have caught the correct change.

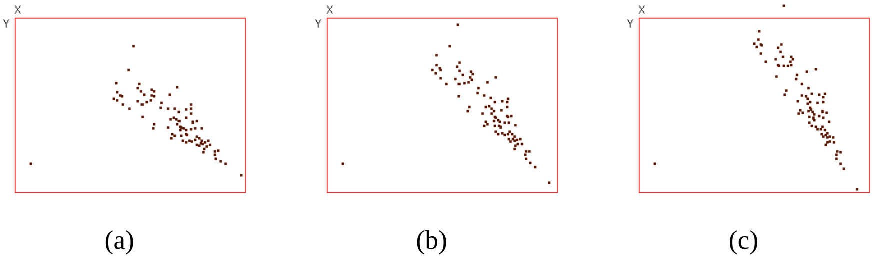
In order to answer these two questions, we performed a user study to observe participants' capabilities in detecting pattern changes on some artificial datasets adapted from real ones. We tested our proposed layout strategies and traditional time-series data visualizations, including line charts and heatmaps. The experiment results can help validate the effectiveness of our proposed techniques, and enable us to derive a guide how to choose layout strategies based on the characteristics of data analysis tasks and datasets. This is useful for both the data analysts as well as visualization system designers.

8.1 Experiment Design

The basic procedure used to design this experiment is as follows: (1) Choose some commonly-used data patterns that can be defined easily and clearly; (2) Construct streaming datasets having changes on selected data patterns between time windows; (3) Generate figures or animations using the proposed visualization techniques, as well as line charts and heatmaps, and design questions to ask participants, regarding the pattern changes in the generated visualizations; (4) Analyze users' response accuracy and response time. In theory, high accuracy and low response time indicates an effective technique. Whether a proposed technique is good depends on many aspects, such as the selected data patterns and the magnitude of pattern change. In this experiment, we tried as many combinations of these factors as possible, and observed how they affected users' responses. Although it is impossible to try all combinations, our experiments aimed to test the most common ones to guide most data analysis tasks.

Choosing Data Patterns: In prior examples, we can see two types of data pattern change: one is the slope change of linear trends (Figures 1, 5), and the other is the movement of the main cluster (Figures 2, 9). They both are very common in many real applications and are easy to explain to participants without experience in visual data analysis. There are some other types of change, such as the displacement of fit lines representing linear trends, the expansion or shrinking of clusters, and changes in other data patterns. Actually, different types of data patterns might be similar to each other, e.g., the displacement and the slope change of a linear trend. Therefore, we may be able to borrow some results on evaluating slope change when we design a system to help users detect displacements in linear trends. If a new data analysis task is totally different from the tested data pattern changes, a new experiment can be designed with this procedure as a guide.

Constructing Datasets: The basic idea for constructing a dataset is as follows: (1) Pick a specific time window, namely W_0, from a real dataset and regard it as the first window of the final experimental data. (2) Construct several artificial time windows, namely from W_1 to W_{n-1}, based on the initial window. We require that the selected pattern is always changing from W_i to W_{i+1} for any i that satisfies $0 < i < n-1$. (3) Generate the final dataset using the windows from W_0 to W_{n-1}. An example dataset is shown in Figure 11. It is generated from a snapshot of traffic data. Figure 11(a) corresponds a subset of the real traffic data, while figures 11(b) and 11(c) are generated using time windows adapted from the data in Figure 11(a).



(a) (b) (c)

Figure 11: Figure (a) - (c) show three time windows of an artificial streaming dataset. Figure (a) is cut and copied from a time window of traffic data. The datapoints in Figures (b) and (c) are constructed from those in Figure (a) via the rotation shown in Figure 12.

The key is step (2). How can we determine the change magnitude? How many time windows can we create? After analyzing some real datasets, we noticed that the change magnitude can significantly affect a user' capability to detect the pattern changes. In addition, if the number of time windows in superimposition is big, users cannot distinguish different windows because of the number of colors and the presence of overlapping, Thus, we created streaming datasets that are combinations of the following two factors:

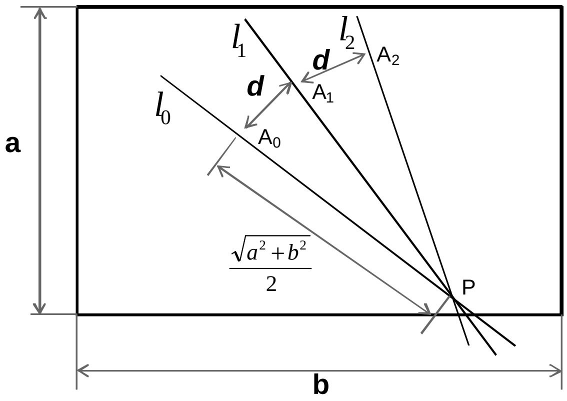


Figure 12: We show how to construct an artificial streaming dataset having three time-windows from a real dataset. The straight line PA_0 represents the linear trend with which the datapoints in a specific time window of traffic data agree. PA_1 and PA_2 correspond to the linear trends of two constructed time windows. Note that we also created an anti-clockwise change and introduce more windows for some questions.

- **Change Magnitude**: Obviously, a bigger change can be more easily perceived by human eyes. We created multiple datasets having various degrees of magnitude for the change in the selected data patterns,
- **The Number of Windows**: We created three types of datasets, with 3, 4, and 5 time windows, respectively.

Figure 12 shows how to determine the magnitude of pattern change (linear trend) between contiguous windows. The straight line PA_0 represents the fit line of the linear trend for the initial window. In this line, the point P is the intersection of fit lines for two contiguous time windows in the real traffic data, one of which is the initial window. For example, if we pick the second time window as the initial window in Figure 1(d), the point P is the intersection of lines A and B. The distance between P and A_0 is half of the diagonal line for the scatterplot. We then construct the fit lines for artificial windows from W_1 to W_{n-1}. In Figure 12, we only show two fit lines $l_1(PA_2)$ and $l_2(PA_2)$ for W_1 and W_2. Note that $|PA_0| = |PA_1| = |PA_2|$ and we use the $d_1 = |A_0A_1|$ and $d_2 = |A_1A_2|$ to represent the change magnitude. In conclusion, in the experiment for linear trend patterns, we use the combination of three types of change magnitude (1, 4 and 12 pixels), and three time window counts (3, 4, 5 windows).

The construction of datasets for cluster motion patterns is similar to the above process.

Generating Visualizations and Questions: In order to make the comparison among the user responses for different techniques meaningful, we follow several rules:

(1) Color Schema: In superimposition and step juxtaposition, the selection of color schema can significantly affect participants' capabilities to detect pattern change. Thus we applied the same color schema to all visualizations generated using superimposition and step juxtaposition. Specifically, we selected a color schema, utilized the colors at the two ends in the step juxtaposition, and chose evenly spaced colors based on the number of time windows for the superimposition.

(2) Canvas Size: Because a small canvas size can lower response accuracy and increase the response time because of possible overlapping, we fixed the total canvas size and assigned a spe-

cific canvas size to each scatterplot based on the layout strategies. To be specific, we allowed the superimposition and animation to use the total canvas size, but put the juxtaposition and step juxtaposition in a grid while maintaining the ratio between width and height for each scatterplot. The total size of the grid is equal to the total canvas size. For example, if we have 5 windows in generating a juxtaposition, we split the total canvas to a grid having $9(3 \times 3)$ cells. This can maintain the shape of scatterplots but utilize the canvas size as much as possible.

(3) Point Size: The point size must be appropriate to convey data patterns in scatterplots since dots that are too small are difficult to distinguish and big dots could result in overlapping. Thus we used 4×4 pixel points in the superimposition and animation, and the 3×3 pixel points in the juxtaposition and step juxtaposition, because the latter have a smaller canvas size.

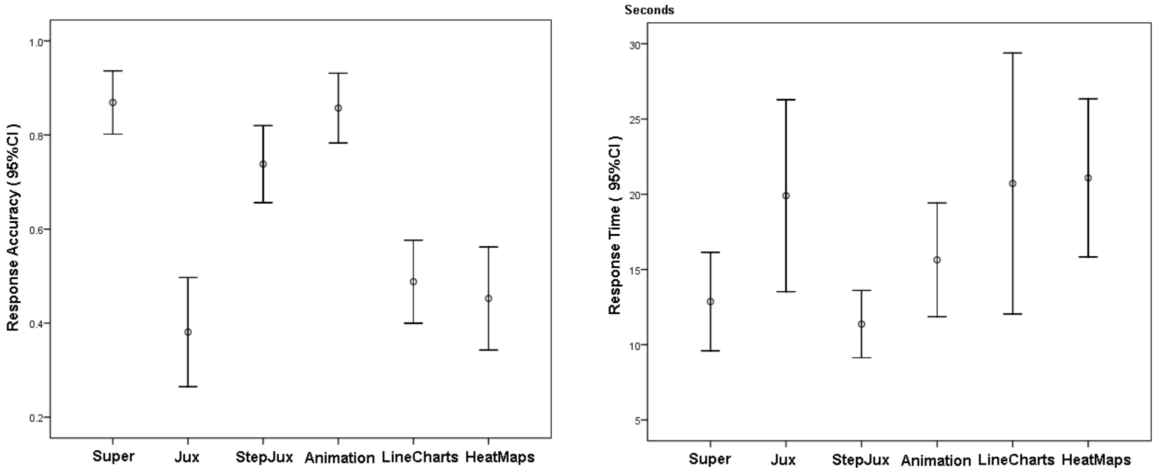
The questions for our proposed techniques are straightforward. For example, in questions about linear trends, we designed a multiple choice question, and directly asked about how the fit line slope changes (increasing or decreasing) between two specific contiguous windows. However, it is almost impossible to perceive a fit line or a cluster in line charts and heatmaps. Thus we gave equivalent questions. For linear trends, we asked participants to estimate the rate of change for one variable with respect to the change of the other, which can be regarded as the fit line slope. We asked users to tell us how this rate changes. In cluster movement questions, we asked users to estimate how the average value on one variable changes from one window to the next.

8.2 Experiment Settings

In total, 14 computer science students attended our experiments. Two of them were undergraduate students, while the others were graduate students. We first gave a short introduction and showed some sample questions to each student, and then asked each to finish two groups of questions. Each group has 33 questions. One group was for linear trends and the other pertained to cluster movement. To avoid the side-effect of a learning curve, we shuffled all questions in each group for each participant. All questions were shown to users via the same laptop.

8.3 Experiment Results

Result 1: Figures 13(a) and 13(b) show the mean values with a 95% confidence interval of response accuracy (RA) and response time (RT) for all participants and questions. We also compared RA and RT values for different visualization techniques using the paired samples t-test and drew the following conclusions: (1) From the aspect of RA, superimposition, step juxtaposition and animation are all significantly better than juxtaposition, line charts and heatmaps ($p < 0.001$). Since every question has only three choices, the performance of juxtaposition, line charts and heatmaps was deemed not acceptable within our experiment configuration because their RA mean values are less than 50%. Thus three of the four proposed techniques can work much better to convey multi-dimensional correlations than traditional time-series data visualizations, such as line charts and heatmaps. (2)Superimposition and animation are a little bit more accurate than step juxtaposition ($p = 0.02$ and 0.05). But superimposition is not significantly different from animation ($p = 0.67$). (3) For the RT values, we only compared acceptable techniques. We can see that participants normally spent less time on step juxtaposition than superimposition and animation. However, superimposition is not significant different from animation($p = 0.10$), while the difference between step juxtaposition and animation is significant ($p = 0.005$). Therefore, for the tasks requiring users' quick response to pattern changes, step juxtaposition is a good option.



(a) (b)

Figure 13: The experiment result for all participants and questions: (a) response accuracy; (b) response time.

Result 2: In order to see how the magnitude of pattern change affects participants' performance, we calculated the mean values with a 95% confidence interval of RA values grouped by the combination of layout strategies and change magnitude. The results are shown in Figure 14, which demonstrates two facts: One, participants have improved performance when the change magnitude (the number of pixels) becomes bigger, except for juxtaposition. The difference between a 1 pixel change and a 4(12) pixel change is significant for all layout strategies except juxtaposition ($p < 0.02$). Moreover, the percentages of correct answers for 4 and 12 pixel change are close to 100% for all layout strategies except juxtaposition. However, the RA values for a 4 pixel change is not significantly worse than those for a 12 pixel change. Two, for the 1 pixel change, animation has the highest RA values, and was significantly better than superimposition and step juxtaposition ($p = 0.04$ and 0.03). Subjects had a correctness percentage of about 65%. Considering that the point size is 4×4 for animation, this is a very good result. The reason is obvious: when the change is very small, the similarity between the datapoints of two windows results in too much overlapping; thus participants cannot perceive subtle changes from the figures we generated. However, animation can avoid the overlapping and still convey the pattern change via the short-term memory of human brain. Therefore, we draw two conclusions: (1) Under our experiment configuration, superimposition, step juxtaposition and animation can work very well for changes bigger than or equal to 4 pixels; (2) Animation can work relatively well even if the change is very tiny and smaller than the point size, while superimposition and step juxtaposition cannot.

An interesting result of this experiment is that juxtaposition is not good at conveying pattern change. We expected that it would at least be better than superimposition because it could relieve visual clutter and make the pattern obvious. Actually, it is not as good as the other three techniques. Our guess is that human eyes cannot detect change from one figure to the other without an appropriate reference if the change is tiny. Recall that we asked participants to detect the slope change

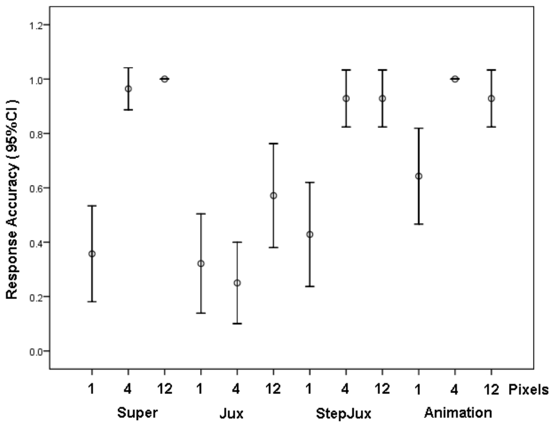


Figure 14: The response accuracy for all participants and datasets having only 3 time windows. The numbers on the horizontal axis mean the change magnitude in the unit of pixels.

of a linear trend. We can observe one time window by treating the datapoints in the other time window as a reference in superimposition, step juxtaposition, and animation, because datapoints of two time windows are put in the same scatterplot. Note that, when an animation shows the second frame, the first frame can still be used as a reference because of human short-term memory. This makes it easy to perceive the pattern change. However, if using juxtaposition, it is difficult to use such a reference because two windows are separated from each other. One possible solution to improve juxtaposition is to add reference points for each individual scatterplot in the form of lines. In such as way, users can easily estimate the parameters for the data patterns in each window, including the slope of fit lines and the distance between a cluster and the scatterplot border. This solution has an obvious disadvantage: grid lines can cause visual clutter and thus counteract their benefits. This should be tested in an experiment, which we plan as future work.

8.4 Evaluation Summary and Implications

We have answered question 1 at the beginning of this section (are these layouts better than traditional techniques) using conclusion 1 of result 1. For question 2, it is difficult to give a simple answer because each technique is appropriate for different cases. Instead, we derive a set of guidelines (Table 3) to advise data analysts and visualization system designers to choose appropriate layout strategies.

The number of windows involved in pattern	The magnitude of the pattern change	
	Small	Large
Small	Animation	Superimposition & step juxtaposition
Large	Animation	Step juxtaposition

Table 3: The guideline to advise data analysts and visualization system designers to choose appropriate layout strategies in terms of the characteristics of datasets and tasks.

We recommend animation when the change is small, because animation is the only technique that appears to work very well in this case. However, animation could make the visualizations annoying and requires a longer response time, so we recommend superimposition and step juxtaposition when the change magnitude is big. Superimposition does not work well when users choose too many windows in the DOI functions, as it can cause bring serious visual clutter and humans cannot readily differentiate many colors at once. In this situation, step juxtaposition is best choice. A key question is: what is this threshold at which superimposition becomes problematic? It is almost impossible to give such a number for all visual analysis tasks, because it depends on many factors, including the selection of color schema, canvas size and the degree of visual clutter. In terms of visual perception theory, human eyes can distinguish at most about 10 colors in one figure. Thus this threshold is 10 or less if datapoints are too crowded in the final visualization. For a specific use case, users or system designers can do an experiment to determine this number.

In Table 3, we divided the change magnitude into small and large. So we must answer a question: what kind of change magnitude can be regarded as small(large)? Let us recall the experiment result shown in Figure 14. Only animation can work relatively well when the pattern change is smaller than the point size. So our recommendation is that if the change is smaller than the size of visual items, we regard it as small and suggest that users use animation to observe the pattern change.

9 Case Studies

In this section, we present three stories using these examples to show the effectiveness of our visualization and interaction techniques. Because we have showed the effectiveness of our proposed visualization techniques in user studies, we will mainly focus on demonstrating the usefulness of DOI functions and their interactions in cases 1 and 2. The former will use a type PP DOI function, while the latter focuses on the trial-and-error exploration process using the DOI function interaction tool. Since all examples used scattplots up to now, we will demonstrate the third case by integrating step juxtaposition into parallel coordinates to show that our proposed layout strategies can be applied to other traditional multivariate visualization techniques.

Case Study 1: Figure 5 uses the measures from the sensor D191 (close to the intersection of I-35W and 35th St.) in the traffic data stream. We use a type PP DOI function as shown in Figure 5 (b). The length of a cycle is one day. The current window is 7:00PM-7:30PM on Tuesday, March 24, 2009. In our implemented system, we used scatterplot matrices to show this example. For the sake of saving space, we chose two interesting subplots from each scatterplot matrix to form Figure 5(a). In the two scatterplots, we can see that *Speed* and *Occupancy* always have a negative relationship with each other. However, the absolute value of the slope for line A is much larger than that for line B. Thus, we can conclude that a small change of *Occupancy* can result in a big change in *Speed*, and vice versa, during the past two sliding windows (6:00PM-7:00PM). This is the result of heavy traffic. Nevertheless, these two dimensions do not obviously affect each other in the current window. A more important finding is that such a change of data patterns did not happen on March 22 and 23, i.e, the traffic from 6:00PM to 7:30PM was not that heavy.

From the scatterplot formed by *Volume* and *Occupancy*, we also can see some interesting change of data patterns. Line C contains only data corresponding to the current window (7:00PM-7:30PM). It shows a positive relationship between these two dimensions. We can also see a negative relationship between *Volume* and *Occupancy* when observing other datapoints with lighter colors (Line D), which denote the time period from 6:00PM-7:00PM. The possible reason is that high occupancy means low speed, which can result in fewer vehicles passing the sensor when traffic is very heavy. Conversely, high occupancy means that more vehicles will pass the sensor (Volume) when traffic is not heavy. A similar change did not happen on March 22 and March 23.

What happened? This is not at traffic peak hours. Finally, from the incident record of Mn/DOT, we discover that flooding happened in the late afternoon on March 24, near the cross of I35W and 42nd St. because of 0.44 inch of precipitation on that day. This is very probably the reason why data patterns changed on March 24, as compared to March 22 and 23.

The analysis of this example confirms that the developed techniques can convey not only the multidimensional correlations for a particular time period, but also the change of data patterns, and even *the change of the changes of data patterns* if using type PP functions.

Case Study 2: Let us investigate Figure 9, which uses the sleep data stream. The current window is 36.5-37 minutes after the beginning of this sleeping experiment. Figures 9(c) and 9(d) use embedded views to convey not only trends for three dimensions but also multidimensional correlations. Figure 9(c) was generated using a DOI function to compare the recent 9 sliding windows as shown in in Figure 9(a). From each scatterplot we can quickly find how the primary data in each sliding window move over time. For example, in the plot with heart rate (*heart*) as X and blood oxygen concentration (*blood*) as Y, we can see that the older data mainly fall into the bottom area, then slowly move in the upper-left direction, and finally return back to a middle position. From line charts, we can find that the heart rate decreased to a minimum value and then slowly went up within the recent sliding windows, and, at the same time, the blood oxygen concentration reached a maximum value and then slowly went down. The findings from line charts are good supplements to conclusions gained via the scatterplot.

In Figure 9(c), we also can see that this interesting change exists in the recent several windows. If we want to see more detail of this change, we can use the DOI function interactive tool to adjust the DOI function for choosing fewer sliding windows to display. The new DOI function is shown in Figure 9(b), which results in a new view (See Figure 9(d)). From the new view, we can more clearly see how the positions of clusters change as compared to Figure 9(c).

If we observe carefully the plot with (*heart*) as X and (*blood*) as Y in Figure 9(d), we can see that some of the data in Window 1 (the window just before the current window) have the lowest positions. We then use data driven brushing to set the query condition for dimensions *heart* and *blood* to select these points (Figure 10). Then the exact bounds for those data satisfying the query on dimension *chest* and timestamp are shown using a shading background. Thus we can discover the time when the movement of clusters happened.

Case Study 3:

This case uses a 5-minute slice of sleep data stream. We split it into 10 time windows and generated Figure 15 using step juxtaposition and parallel coordinates. Let us first observe the changes of the correlation between the heart rate and blood oxygen concentrate. Basically, we can see two types of distribution: Type 1: low heart rate and high blood oxygen concentrate, such as windows 54.0-54.5, 56.5-57.0, and 58.0-58.5; and Type 2: high heart rate and low blood oxygen concentrate, e.g., windows 54.5-55.0 and 57.0-57.5. The datapoints in some windows are the mixture of two types of data, such as window 57.5-58.0. In each sub-figure, we can clearly see how the data is changing from one type to the other. For example, in figure 15(a), the data changed from type 1 to type 2. We investigated the whole stream and found that type 1 is the primary one and type 2 concentrates in some parts of the stream. Thus type 2 can be treated as an outlier. Since the patient in this sleeping experiment shows sleep apnea (periods during which he takes a few quick breaths and then stops breathing for up to 45 seconds), type 2 data might be associated with this abnormality. Is this guess true? The third dimension, the chest volume, that is the indicator of respiration, can tell us the answer. During the normal human breath, the chest volume should change in a waving way. We can see that in most of time windows, chest volume values exist in a wide range, which is normal. However, we can find that the values of this dimension in four windows have a very narrow range, including windows 54.0-54.5, 55.5-56.0, 56.5-57.0, and 58.0-58.5, where the patient might stop breathing for a while. Moreover, just after each of these four windows, we can see that data changed from type 1 to type 2. For example, in sub-figure (c), the darker points correspond to window 55.5-56.0, where we guess the patient stopped breathing. We can find that the data changed from type 1 to type 2 in sub-figure (d). Thus, we have a finding via the help of visualizations, that the patient will change to an abnormal condition of high heart rate and low blood oxygen concentrate after sleep apnea happened. We have not confirmed this finding with medical expert, but this case study at leat can show that the proposed layout strategies is so powerful that it can help us find possible cause and effect in data streams, when we apply them to traditional multivariate visualizations. This can help promote hypotheses and confirm new findings.

10 Conclusions and Future Work

In this paper, we have proposed a framework for the exploration of multivariate data streams. The story starts from a basic idea to display both the current data and abstractions of past data to show not only the data patterns in a particular time period but also how data changes over time. We split the whole stream into non-overlapped sliding windows and apply uniform sampling to each window. The sampling ratio for a particular window is determined by a DOI (degree of interest) function to reflect users' interest. A larger DOI value results in a larger sampling ratio for the specified window, which shows more detail in the final view. We provide two types of DOI functions to satisfy some common data analysis tasks, as well as a DOI function interactive tool to allow users to adjust the DOI function when exploring data streams. In order to show how data patterns change, we have proposed four layout strategies, superimposition, juxtaposition, step juxtaposition and animation, to place sliding windows. The evaluation showed that three of these can effectively convey the multivariate pattern change compared to the traditional time-series data visualization techniques. We also derived a guide to advise data analysts and visualization system developers in choosing appropriate layout strategies based on the characteristics of datasets and data analysis tasks. In addition, we allow users to use multiple views in the final visualization and support linked brushing to highlight a subset of interest in all views when users define a query in one view.

Some potential future work includes:

- The current framework is totally user-driven. Most parameters, including those within DOI functions and the length of sliding windows, are controlled by users. In the future, we plan to extend the current framework with a data-driven feature, to enable the system to automatically adjust DOI functions and other arguments in terms of data features. For example, when data patterns change quickly, the system should be able to automatically shorten the sliding window to facilitate the observation of changes.

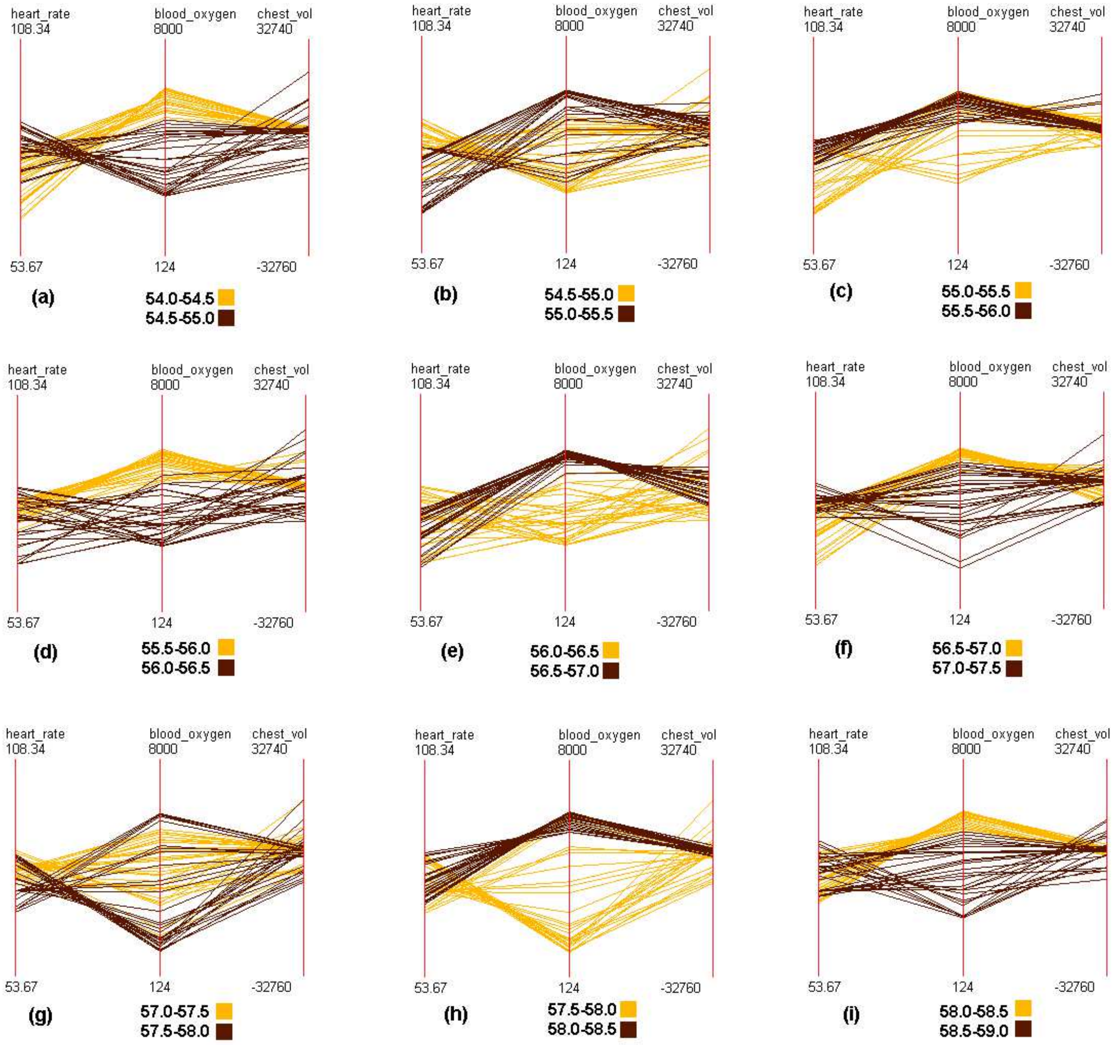


Figure 15: The visualization for a slice of sleep data stream generated by applying step juxtaposition to parallel coordinates. All time units in the figure are minutes. We can clearly see how the relationship between variables changes from one window to the next.

- In this paper, the detection of pattern change depends on visual analysis. For instance, users must compare the shape of the datapoints of two windows to detect whether the slope of the fit line is changing. In the next step, we plan to integrate statistics and data mining algorithms to compute the pattern changes, and directly visualize the change, using either juxtaposition or superimposition strategies.

11 Acknowledgements

This work is supported under NSF grants CCF-0811510 and IIS-0812027. Special thanks go to Di Yang, Zhenyu Guo, and Abhishek Mukherji who provided assistance of many forms throughout this research.

References

[1] C. Albrecht-Buehler, B. Watson, and D. A. Shamma. Visualizing live text streams using motion and temporal pooling. *IEEE Computer Graphics and Applications*, 25(3):52–59, 2005.

[2] A. Arasu, B. Babcock, S. Babu, M. Datar, K. Ito, I. Nishizawa, J. Rosenstein, and J. Widom. Stream: The stanford stream data manager. In *SIGMOD Conference*, page 665, 2003.

[3] L. Berry and T. Munzner. Binx: Dynamic exploration of time series datasets across aggregation levels. *IEEE Symp. Information Visualization Poster*, page 215.2, 2004.

[4] Q. Cui, M. Ward, and E. Rundensteiner. Enhancing scatterplot matrices for data with ordering or spatial attributes. *Visualization and Data Analysis, Part of IS&T/SPIE Symposium on Electronic Imaging*, pages 0R1–0R11, 2006.

[5] G. Furnas. Generalized fisheye views. *Proc. ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 16–23, 1986.

[6] L. Golab and M. T. Özsu. Issues in data stream management. *SIGMOD Rec.*, 32(2):5–14, 2003.

[7] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000 (June 13).

[8] M. C. Hao, U. Dayal, D. A. Keim, D. Morent, and J. Schneidewind. Intelligent visual analytics queries. *Proc. IEEE Symp. Visual Analytics Science and Technology*, pages 91–98, 2007.

[9] M. C. Hao, U. Dayal, D. A. Keim, and T. Schreck. Importance-driven visualization layouts for large time series data. *Proc. IEEE Symp. Information Visualization*, pages 203–210, 2005.

[10] M. C. Hao, U. Dayal, D. A. Keim, and T. Schreck. Multi-resolution techniques for visual exploration of large time-series data. *EuroVis07: Joint Eurographics - IEEE VGTC Symp. on Visualization*, pages 27–34, 2007.

[11] M. C. Hao, D. A. Keim, U. Dayal, D. Oelke, and C. Tremblay. Density displays for data stream monitoring. *Comput. Graph. Forum*, 27(3):895–902, 2008.

[12] B. Lichtenbelt, R. Crane, and S. Naqui. *Introduction to Volume Rendering*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1998.

[13] R. G. Lyons. *Understanding Digital Signal Processing*. Prentice Hall PTR, Upper Saddle River, NJ, USA, second edition, 2004.

[14] A. Martin and M. Ward. High dimensional brushing for interactive exploration of multivariate data. *Proc. IEEE Visualization*, pages 271–278, 1995.

[15] S. Miksch, W. Horn, C. Popow, and F. Paky. Utilizing temporal data abstraction for data validation and therapy planning for artificially ventilated newborn infants. *Artificial Intelligence in Medicine*, 8(6):543–576, 1996.

[16] Minnesota Department of Transportation. Mn/DOT traveler information. http://www.dot.state.mn.us/tmc/trafficinfo/, accessed on Feb. 25, 2009.

[17] J. F. Roddick and M. Spiliopoulou. A survey of temporal knowledge discovery paradigms and methods. *IEEE Trans. Knowl. Data Eng.*, 14(4):750–767, 2002.

[18] B. Shneiderman. The eyes have it: a task by data type taxonomy for information visualization. *Proc. IEEE Symposium on Visual Languages*, pages 336–343, 1996.

[19] D. F. Swayne, D. Temple Lang, A. Buja, and D. Cook. GGobi: evolving from XGobi into an extensible framework for interactive data visualization. *Computational Statistics & Data Analysis*, 43:423–444, 2003.

[20] E. Tufte. *Envisioning Information*. Graphics Press, Cheshire, CT, USA, 1990.

[21] C. Ware. *Information Visualization, Second Edition: Perception for Design*. Morgan Kaufman, San Fransisco, CA, USA, 2004.

[22] P. Wong, H. Foote, D. Adams, W. Cowley, and J. Thomas. Dynamic visualization of transient data streams. *Proc. IEEE Symposium on Information Visualization*, pages 97–104, 2003.

[23] Z. Xie, S. Huang, M. O. Ward, and E. A. Rundensteiner. Exploratory visualization of multivariate data with variable quality. *Proc. IEEE Symposium on Visual Analytics Science and Technology*, pages 183–190, 2006.

[24] C. Yu, Y. Zhong, T. Smith, I. Park, and W. Huang. Visual mining of multimedia data for social and behavioral studies. In *Proc. IEEE Symposium on Visual Analytics Science and Technology*, pages 155–162, 2008.