Exploring Large Scale Time-series Data Using Nested Timelines

Zaixian Xie[a] and Matthew O. Ward[b] and Elke A. Rundensteiner [b]

[a]Oracle America Inc., 1 Oracle Drive, Nashua, NH, USA
[b] Worcester Polytechnic Institute, 100 Institute Road, Worcester, MA, USA

ABSTRACT

When data analysts study time-series data, an important task is to discover how data patterns change over time. If the dataset is very large, this task becomes challenging. Researchers have developed many visualization techniques to help address this problem. However, little work has been done regarding the changes of multivariate patterns, such as linear trends and clusters, on time-series data. In this paper, we describe a set of history views to fill this gap. This technique works under two modes: merge and non-merge. For the merge mode, merge algorithms were applied to selected time windows to generate a change-based hierarchy. Contiguous time windows having similar patterns are merged first. Users can choose different levels of merging with the tradeoff between more details in the data and less visual clutter in the visualizations. In the non-merge mode, the framework can use natural hierarchical time units or one defined by domain experts to represent timelines. This can help users navigate across long time periods. Grid-based views were designed to provide a compact overview for the history data. In addition, MDS pattern starfields and distance maps were developed to enable users to quickly investigate the degree of pattern similarity among different time periods. The usability evaluation demonstrated that most participants could understand the concepts of the history views correctly and finished assigned tasks with a high accuracy and relatively fast response time.

Keywords: Time-series data, multivariate data, visual analysis, timeline.

1. INTRODUCTION

Advances in hardware enable people to record massive data in terabytes or petabytes. Time-series data is an important subtype of routinely collected data .[1,2] On these datasets, analysts often need to discovery data patterns and their changes over time to explain existing phenomena and do appropriate prediction. The data volume is a huge challenge to achieving these analysis tasks.

In recent years, people have agreed that visualization can play a critical role in the processes of data analysis and decision-making, since it allows analysts to use visual perception to uncover important patterns, such as clusters, associations, relationships, and trends. Moreover, visual analytics can provide an interactive environment that combines human visual cognitive capabilities with high performance computations, thus improving the speed and accuracy at which analysts discover data patterns. The visualization community has developed many techniques to help uncover and monitor the data patterns in large scale time-series data. However, there has been little work on visualizing the changes on multivariate patterns for time-series data.

For the above task, we must develop visualization techniques to represent timelines for a relatively long time range and convey the changes of multivariate patterns. For this, we borrow the idea of distorted timelines that can be found in recent literature .[3–6] In these techniques, multiple timelines are used to allow users to easily navigate to an arbitrary time period.

As a motivating example, we introduce a traffic dataset provided by Mn/DOT (Minnesota Department of Transportation).[7] Mn/DOT installed more than one thousand sensors on highway entrance/exit ramps and main lanes throughout the Twin Cities Metro area. Each detector collects a value for each of the following three 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; and (3) Speed: the average speed of all vehicles passing the detector. We chose the collected data in 2008 for sensor D722, and only focus on two dimensions to investigate their correlations. One dimension is the average vehicle speed (*Speed*), and the other is the percentage of time that the detector sensed a vehicle (*Occupancy*). Figure 1 shows an example of hierarchical timelines having five levels. Users can specify two levels: perspective and pattern. On

Further author information: (Send correspondence to Zaixian Xie)
Zaixian Xie: E-mail: zaixian.xie@oracle.com Project Homepage: http://davis.wpi.edu/~xmdv

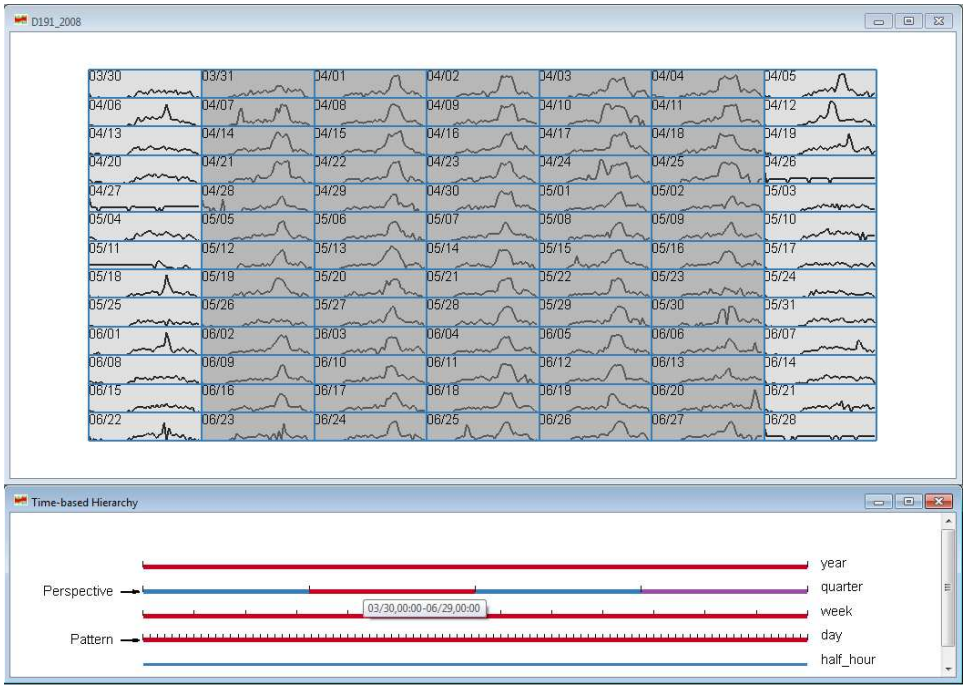

Figure 1. This figure shows a history view (top) with a hierarchical time structure (bottom) defined by users. In time-based hierarchy, all segments are rendered in two colors, blue and purple, alternatively, to help users distinguish them. We focus on the changes across contiguous windows on the pattern level (days) in this figure. The selected quarter (March 30 - June 29) on the perspective level is highlighted in red and indicates the time periods of interest. The red on the week and day level means all segments in the selected quarter are chosen. Grey background is applied to all weekdays in the history view (top) to help readers focus their attention. Note that, in the history view, each glyph (plot) corresponds to one day (pattern level) and contains a curve to represent the slope change of regression lines within 48 time windows for each day.

the perspective level, users can select a time range within which they can observe the data pattern changes. The time window on the pattern level is the smallest time unit on which users can uncover the data patterns. In Figure 1, the user has specified a perspective level on quarters and a pattern level on days. It means that the user wanted to investigate how the traffic patterns change across days within a selected quarter. Figure 1 also shows a history view composed of glyphs in a grid. Each glyph is a small plot corresponding to one day in the selected quarter and conveys the slope change of the regression line (*Occupancy* against *Speed*) via a curve. Each row is a week, and the columns are the days of the week.

This technique makes it easy for users to discover cyclic pattern change phenomena on the time-series datasets having hierarchical time structure. In addition, if we apply more visualization and interaction techniques to it, we can easily investigate the similarity among time windows. For example, distance measures can be used to represent the differences among glyphs, which can be mapped to colors.

When users explore the traffic data, they may want to observe the detailed information within one day instead of pattern abstractions. In other words, they want to observe the datapoints themselves instead of pattern vectors. One intuitive solution is to display one scatterplot for each time window. For this, we need to draw 48 scatterplots for one day. Clearly, this needs too much canvas space. In this paper, we instead use the merge algorithm developed by Xie et al.[8] to reduce the number of visualized windows in each day, generating a hierarchical structure to allow users to select an appropriate level to observe how data patterns change in one day.

The main contributions of this paper include:

- A framework was designed to convey how time-series patterns change over a relatively long time period. The main idea is to generate a hierarchical structure for timelines, with which users can easily navigate within the data.
- Dimensional reduction and similarity-based selection techniques were applied to the time-series data for simplifying the exploration on the similarity among time windows.

- A usability evaluation was performed to confirm that most users can correctly understand the concepts involved in these new techniques and can learn to use the implemented system with little difficulty.

2. FRAMEWORK

Definitions

Some terms used in this framework are given as follows:

Pattern level: A level in the time hierarchy on which a window is a basic unit for users to observe data patterns during pattern evolution. For example, if users want to investigate how traffic patterns change from one day to another, the "day" is the pattern level.

Pattern window: A time window on the pattern level.

Pattern range: The time range on the pattern level containing pattern windows among which users want to explore the traffic pattern changes.

Perspective level: The highest level on which users can select one or more time windows to define the time range containing pattern windows of interest. For example, if users are interested in the traffic pattern changes across days within one quarter, the perspective level is "quarter".

Perspective window: A time window on the perspective level.

Perspective range: The selected time range on the perspective level.

Figure 2 shows how to generate history views. This framework assumes that the time-series data can be defined using a hierarchical structure. On each level, users can define a time unit, and then the time-series data is split into many segments. One segment at a specific level could contain several segments at the lower level. We call such segment a *time window*. For example, traffic can be studied at five levels, including year, quarter, week, day and half hour. One year contains 4 quarters, each of which has 13 weeks, and so on. This hierarchical structure is shown at the left side of Figure 2. It has n levels. For the traffic data, $n = 5$. Each segment at levels L_0, L_1, L_2, L_3 and L_4 corresponds to a half hour, day, week, quarter, and year, respectively.

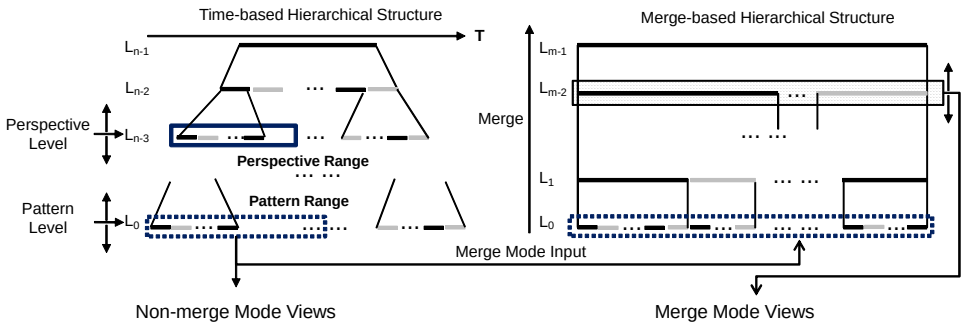

Figure 2. The framework to generate history views using nested timelines. The left side shows hierarchical time units that contain two levels, perspective and pattern. At both levels, users can specify time ranges, named perspective and pattern ranges. All time windows in the pattern range can be directly output to a non-merge mode view, or the merge algorithm can be used to generate a merge-based hierarchical structure. Users can select a specific level on this structure. All time windows on this level will be output to a merge mode view.

On this user-defined hierarchical structure, users can specify a *pattern level* and a *perspective level*. The former indicates the time unit in which users want to observe the data patterns. Similarly, an arbitrary time range on the latter level, is called the *perspective range* (highlighted by a blue solid line rectangle at the perspective level). The blue dashed line rectangle at the pattern level contains all time periods (*pattern range*) over which users want to study pattern changes. For instance, imagine that a user moves the perspective level to week, and the pattern level to half hour, and then selects

a specific week. Thus the pattern range contains all time windows (half hours) within this week. Then, in this case, users focus on investigating how data patterns change across these time windows within this week.

A key task is to visually convey the pattern changes within the pattern range. To solve this problem, two approaches, named non-merge and merge modes, were designed. For the non-merge mode, we generate the visualizations for each time period and organize them on the history views called "non-merge mode views", i.e., mapping one time window on the pattern level to one glyph. This mode is applicable for conveying how data patterns change across the entire perspective range. However, if we want to study how data patterns change within one pattern window, we need some other techniques. Our solution is merged mode. In this mode, we apply the merge algorithm described in[8] to all time windows in one pattern range. The basic idea is to compress multiple time windows to one if their data patterns are similar. Thus, we can use less canvas space to visualize the changes of data patterns on many time windows. This algorithm works in a hierarchical way via multiple rounds of merging. A threshold δ_i is given in Round i ($\delta_1 < \delta_2 < \delta_3 < ...$). In Round 1, we repeatedly scan time windows, and merge two contiguous windows to one if the pattern difference between them is less than δ_1. Now, we can get a list L_1 having fewer windows than the original window list. Then we repeat this process but use δ_2 as the threshold. Thus we can get another window list having fewer windows than L_1. We repeatedly run the merge algorithm until we get a list having only one window. For example, in Figure 3, we apply this merge algorithm to the time windows on April 18, 2008, and repeat the merge algorithm 6 times, resulting in a merge-based hierarchy (bottom right in Figure 3) having 7 levels.

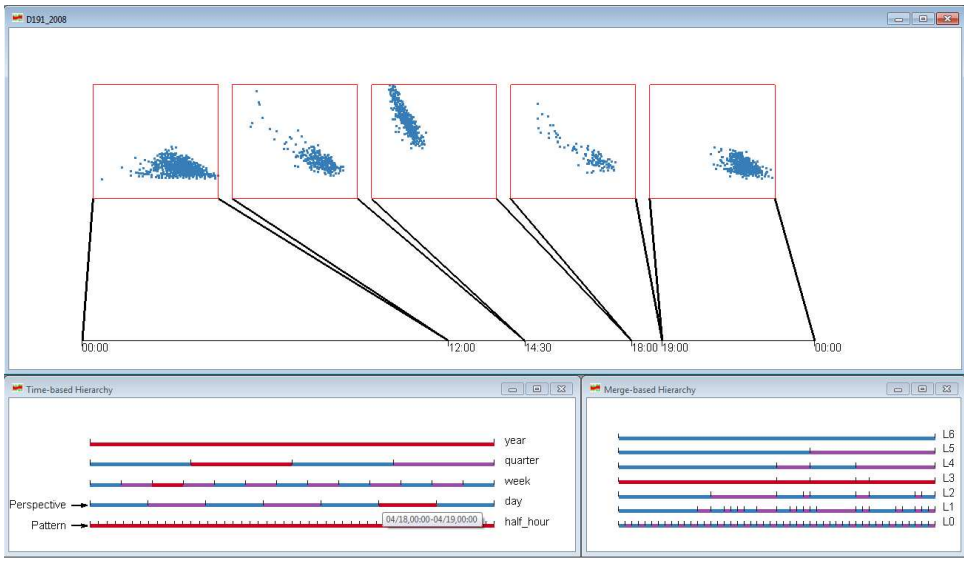

Figure 3. A snapshot of history views showing merged mode views: one day of data is selected in the left hierarchy, and the merge mode view show the result of the third round of merging.

Figure 3 shows a snapshot of the implemented visualization system based on the above proposed framework under the merged mode. This system is composed of three views: a time-based hierarchy (bottom left), a merge-based hierarchy (bottom right), and a history view (upper section). The first two views correspond to the time-based hierarchical structure and the merge-based hierarchical view in Figure 2, respectively. The history view can be non-merged views or merged-views based on the user's selection. A merged history view is shown in Figure 3. In the time-based hierarchy view, one timeline is shown for each level in this dataset. Thus this hierarchy has five timelines, corresponding to year, quarter, week, day and half hour. Users can use the mouse to drag the perspective and pattern level tag to change them. Note that in this case the perspective level is set to the day and the pattern level is on the half hour. Since the data is only for one year, the top level has only one segment that is always selected and highlighted in red. Four segments on the second level at the bottom left part of Figure 3 correspond to four quarters in this year. For instance, the second segment is from April 1 to June 30. Users can click one segment to select this quarter and highlight it in red. The timeline in the week level contains the thirteen weeks in this quarter. That means the first segment is the week from April 1 to April 7, and the second one corresponds to the following week (April 8 - 14). Users can continue to select one week or more on the week level timeline, and then do similar things on the following levels, until reaching the perspective level, the day timeline. If one

day on this level is highlighted, all time windows on the pattern level (half hour) will be highlighted. In the bottom left part of Figure 3, April 18 is selected at the perspective level. Then, the highlighted time windows (half hours, i.e., pattern level) in this day are highlighted and output to the merge algorithm for generating the merge-based hierarchy (the bottom right section of Figure 3). In this hierarchy, users can select and highlight a whole level instead of one segment, which is different from the time-based hierarchy. Then all merged windows on the selected level will be visualized in the history view at the upper section of Figure 3.

3. VISUALIZATION TECHNIQUES FOR MERGED MODE

To help users understand how time windows are merged from one level to the next, an approach, named the *two levels view*, was designed to display the merged windows on two levels together. An example is shown in Figure 4. In this figure, two levels, "L2" and "L3", are selected in the merge-based hierarchy. The corresponding time windows in these two levels are displayed in the history view simultaneously. The two levels are both connected to the same time axis. From this figure, one can clearly see how time windows are merged from one level to the next.

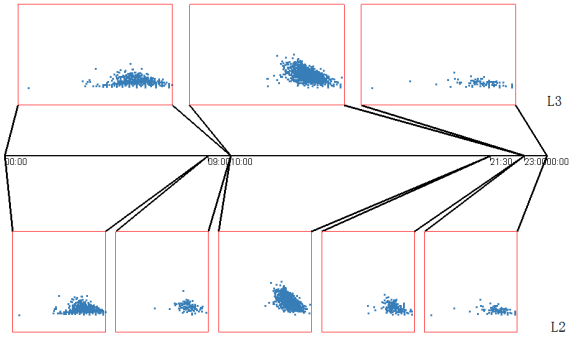

Figure 4. This figure shows the merged mode with the *two levels view*. Two levels (L2 and L3 in Figure 3) on April 20, 2008 are selected to show how windows are merged from L2 to L3.

In conclusion, merged mode views can help users perform the following data analysis tasks in the history data:

- Observe the data pattern changes at the pattern level. Users can adjust the number of displayed windows in terms of canvas size and the degree of visual clutter.
- Investigate how time windows are merged from one level to another via the *two levels view*.

4. VISUALIZATION TECHNIQUES FOR NON-MERGED MODE

Figure 1 shows a non-merged grid view. It can help analysts study how data patterns change across a relatively long time range. However, It has some obvious disadvantages: (1) This technique cannot work for the case where the difference between the perspective level and pattern level is bigger than 2 since it is a two dimensional grid-based visualization technique; (2) It does not explicitly convey the pattern change from one day to the next. In this section, extensions will be applied to the proposed approach for solving the above problems. Section 4.1 focuses on issue 1; Section 4.2 proposes visualization and interaction techniques for explicitly representing the pattern changes.

4.1 Virtual Calendar View

In many cases, users may want to study the pattern changes over a relatively long time range. For example, in the traffic data, a common analysis task is to observe how patterns for each day change across one year. Since the time hierarchy has five levels (year, quarter, week, day and half hour), the perspective level should be year, and the pattern level is day. The difference between these two levels is 3, so the non-merge model views described in Section 2 does not work unless a certain extension is applied to it. For this data analysis task, the solution is to render one grid view for each quarter, and then generate the final visualization by laying out four views horizontally (Figure 5). Note that this is similar to a calendar, thus it can be called a *virtual calendar view*. The reason why to call it *virtual* is that this approach can be used on the data

having a hierarchical structure which actually is not based on natural calendar units (year, quarter, week and so on). In such a case, the view is not a real calendar.

The above approach was inspired by Wijk and Selow's calendar view [9] and *MulteeSum* [10] developed by Meyer et al. Wijk and Selow used a real calendar to visualize the numbers of employees present at a research center that were encoded by colors. Meyer et al. visually represented the gene expression profiles of cells via a small-multiple matrix of line charts. Each row corresponds to a cell while each column is a gene. Then one line chart can convey the time series data for the expression of one gene in a specific cell. Our non-merge mode views are very similar to *MulteeSum*, but rows, columns, and glyphs all are representing different time units in a hierarchical way.

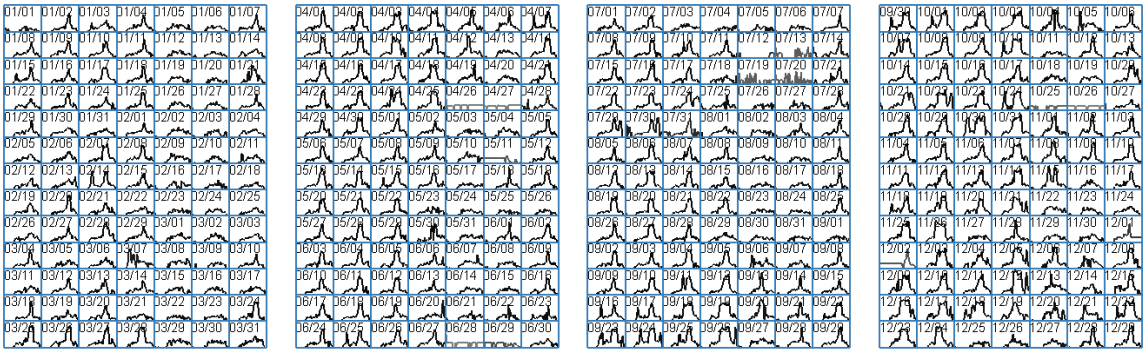

Figure 5. 2D grid view is extended to create a *virtual calendar view*.

For our extension, we added a new level, called the calendar level, between perspective and pattern level in Figure 2. It is two more layers higher than the pattern level. If users selected a range in the perspective level, it will contain some continuous segments on the calendar level, which can be called the *calendar range*. For each segment in this range, a grid view is generated. In Figure 5, each segment on the grid level corresponds to one quarter. All grid views are then organized horizontally and vertically in a bigger grid, or via other layout strategies to obtain the final visualization. Theoretically, this approach can work regardless of the number of segments in the calendar range. However, if this number is too big, the quality of the final output will be dramatically reduced because the plots become too small. Therefore, in real applications, users should select an appropriate number of segments on the grid level so as to avoid low quality output.

4.2 Explicitly Conveying Pattern Changes

In all the above visualizations, the patterns for each segment in the pattern level were conveyed to the users. It is true that data analysts can investigate how patterns change across the selected time range by observing the entire figure and comparing glyph shapes. However, this is time consuming especially if there are hundreds of glyphs in the final output. In this section, two visualization techniques that can explicitly convey the pattern changes will be discussed: *MDS pattern starfield* and *distance map*. Then some interaction techniques based on them will be introduced.

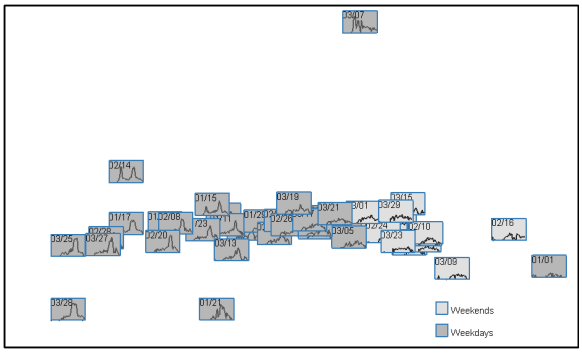

Figure 6. MDS algorithm is used to generate positions for cells. The distance between cells represents the distance between corresponding curves.

MDS Pattern Starfield

Multidimensional scaling (or MDS)[11] is a commonly used approach in data visualization to convey the distance among multiple objects. For example, Yang et al. developed the MDS VaR display to visually represent the distances among multiple dimensions in a large-scale multivariate dataset .[12,13] Figure 6 shows an MDS pattern starfield to present a pattern space for the first quarter in the traffic data. Similar to Figure 1, each glyph corresponds to one day, i.e., one pattern window. The proximity among glyph positions reflects pattern distances. Assume that users want to observe N pattern windows, the procedure to get such a starfield is as follows: (1) Calculate distances among these pattern windows and record them in an N $\times$ N matrix. (2) Treat this matrix as the input to an MDS algorithm,[11] reducing the dimensionality to 2 to generate glyph positions. (3) Render each pattern window as a glyph in the position obtained from the MDS algorithm.

The advantage of this approach is obvious. First, users can easily observe the distribution of pattern windows and clusters in pattern space, since this layout conveys the distance among pattern windows. Then different actions, such as manual clustering and outlier detection, can be easily applied to pattern windows. For example, in Figure 6, one can see that there are several outliers: the glyphs corresponding to the pattern windows on Jan. 1, Jan. 21, Feb. 14, Feb. 16 and March 7. In addition, one interesting phenomena is that weekends mainly occupy the right part of the figure while weekdays are in the middle and left sections. This is easy to explain because weekend traffic patterns are significantly different from those in weekdays. In fact, it is expected that two clusters will form, one representing weekdays, and the other being weekends. Due to some outliers, these two clusters are not clearly separated.

To avoid the impact of outliers and observe whether two clusters (weekends and weekdays) exist in this dataset, we introduced an interaction technique to allow data analysts to remove some glyphs from the figures. When users move the mouse to a specific glyph, they can right click this glyph, it will be removed from the input of the MDS algorithm. In this case, it will not be rendered in the final output. Users can repeat this action multiple times to remove multiple glyphs. Using this technique, two glyphs, March 7 and Jan. 1, were removed from Figure 6. The results are shown in Figure 7. Now, two clusters can be seen clearly.

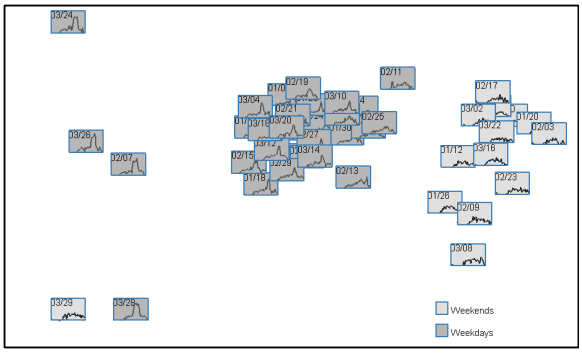

Figure 7. Two days, March 7 and Jan. 1, were removed from Figure 6, since they are obvious outliers. Now one can clearly see two clusters: weekdays and weekends, along with a small set of outliers.

Distance Map

The main goal of the distance map is to convey the pattern distance among pattern windows. Assume users selected N pattern windows in the *Pattern Range*, there are N$\times$N possible distance measures that can be represented. It is not practical to show all these measures in one figure. For example, there are $364 \times 364 = 132496$ distance measures in Figure 5. Actually, in most data analysis tasks, users often are only interested in the distance between two specific pattern windows, or between one target window and all others. A typical scenario is as follows: the data analyst finds one interesting pattern window, and then wants to investigate how this window is different from others, or how patterns change around this window. Thus what needs to be shown is the distance measures between this target window and its neighbors. Therefore, we allow users to specify a target pattern window, and then use the distance map to show the pattern distance measures between this target window and others.

The other problem is how to visualize the distance measures. Since the color is a visual variable that has a high degree of preattentive processing,[14,15] an encoding technique is employed to map distance measures to glyph background colors. The generated output is called a *distance map*. An example is shown in Figure 8. In this figure, Feb. 3 is selected as the

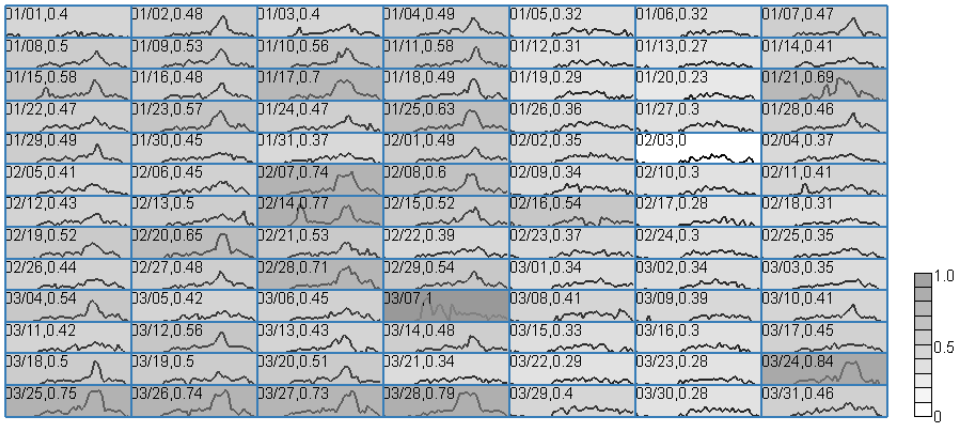

Figure 8. This figure shows an example of the *distance map*. One day (Feb. 3) is selected as the target pattern window. The distance measures between other pattern windows and this day are shown in each grid and explicitly represented by the background color. The legend shows how the distance measures are mapped to colors.

target pattern window. The implemented visualization system calculates the distance measures between this target windows and all others, and then shows all measures via the glyph background color. All distance measures are normalized before visualizing via the following formula:

$$d' = \frac{d - d_{min}}{d_{max} - d_{min}}$$

where d_{max} and d_{min} are the maximal and minimal distance. d is the real distance and d' is the normalized value. For the examples in this section, $d_{min} = 0$, because the distance between one pattern window and itself is 0. All distance measures are positive values. Obviously, the biggest distance measure will be normalized to 1. This distance is also shown in each glyph. For example, in the glyph for Jan. 1, the number 0.4 is the normalized distance measure between Jan. 1 and Feb. 3. One can also find that March 7 has a normalized distance equal to 1 (maximal possible value), therefore, its glyph has the darkest color.

In Figure 8, one finding is that most glyphs in the fifth column (Saturdays) and the sixth column (Sundays) have a smaller distance to Feb. 3 (Sunday) since their background colors are brighter than other columns. In other words, the weekday columns (the first to the fourth and the last) have a smaller similarity to Feb. 3 than the Sunday and Saturday columns. This observation is consistent with common sense that normally the traffic patterns during weekends are significantly different from those on weekdays.

5. USABILITY EVALUATION

In the prior sections, we showed the proposed history views under merged and non-merged modes along with their associated interaction techniques. Many examples have demonstrated that the proposed techniques can help data analysts efficiently discover how data patterns change across pattern windows in different hierarchical levels over a very long time period. However, the discussion and examples in the prior sections are not sufficient to show the usability of the implemented system based on the proposed framework. A pending question is: can users understand concepts about the proposed data model and visualization/interaction techniques and learn to use this system? To answer this question, we designed a usability evaluation and invited some subjects to use the implemented system for performing some specific data analysis tasks.

The basic idea to design this experiment is as follows: (1) Design some data analysis tasks on time-series data related to the above questions; (2) Invite participants to perform these tasks; (3) Record the average response time and response accuracy for each task.

In this experiment, we continue to use the traffic data in prior sections, since it has many features, such as cyclic changes and outliers, which we wanted to ask participants to look for. Eleven software engineers from Microsoft attended this experiment. All of them had experiences with simple visualizations, including line charts, bar charts and scatterplots,

Task Type	No.	Response Time (Seconds)		Response Accuracy	
		Avg	Std Dev	Avg	Std Dev
Timelines	1	8.3	3.8	1.0	0.0
	2	11.3	3.6	1.0	0.0
Merge Mode	1	130.8	14.9	0.86	0.23
	2	160.5	29.7	0.82	0.25
Grid Views	1	197.0	45.9	1.0	0.0
	2	168.0	54.2	0.95	0.15
MDS	1	21.7	10.1	0.95	0.15
	2	33.0	15.7	0.91	0.20
Distance Map	1	113.0	41.0	0.73	0.34
	2	121.5	41.4	0.77	0.26

Table 1. The response times fro the usability experiment.

but had not worked with some advanced ones, such as the MDS layout. Participants first received training to familiarize themselves with out system, and then performed 20 tasks belonging to 10 categories as below:

Timelines (Navigating Timelines)

1. Find a specific day in the time-based hierarchy.
2. Find a specific half hour in the time-based hierarchy.

Merge Mode (Investigating Merge Mode Views)

1. Find where the biggest change of the regression line slope is in a specific day.
2. Find up to three biggest changes between adjacent time windows in the regression line slope in a specific day.

Grid Views (Browsing Grid and Virtual Calendar Views):

1. Observe the traffic pattern trends within a specific quarter using the grid views and choose the correct description from multiple choices.
2. Observe the traffic pattern trends within one year using the virtual calendar views and choose the correct description from multiple choices.

MDS (Understanding MDS Pattern Starfield)

1. Look for the outliers or clusters, if any, in an MDS pattern starfield.
2. Remove one or more outliers until clusters can be clearly seen.

Distance Map (Mastering Distance Map)

1. Find the top 10 glyphs closest to a specific day regarding the traffic pattern trends using the distance map. If users can find more than 7 correct glyphs, the answer was treated as correct.
2. Find the top 10 glyphs farthest from a specific day regarding the traffic pattern trends using the distance map. The standard for correct answers was the same as the previous task.

Table 1 lists the response time and response accuracy for these 12 tasks.

From these experiments, the conclusions are:

- Users easily understood the time-based and merge-based hierarchy.
- The merge-based hierarchy is effective in helping users browse a short time period with uneven pattern change rates. Most users located the pattern changes correctly.
- Users retrieved pattern trends from the grid views and the virtual calendar views with relatively ease. Although the tasks normally took them about three minutes on average, the response accuracy is very high.
- The MDS pattern starfield conveyed the clusters and outliers in pattern windows very well. Users quickly learned how to remove outliers from the view and make the clusters separate.
- Two types of tasks in "Distance Map" have a low but acceptable response accuracy. It shows that the distance map achieved the goal to represent the pattern distance among time windows, but needs improvement. Most users complained that it is difficult to distinguish the background color for glyphs, so a better color scheme is necessary.

6. RELATED WORK

Time-series data has been identified as one of basic data types[16] in the area of information visualization. Appropriate timelines are necessary and important to visualize time-series data. An intuitive approach to time-series data layout is timelines,[17] which use the horizontal axis to represent time. Other than timelines, researchers have developed other layout methods to facilitate specific analytic tasks. *Spiral Graphs*[18–21] use a spirally shaped time axis to visualize temporal data having seasonal cyclic characteristics.

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 algorithms can be categorized into two approaches: data-driven[22] and user-driven.[4,23] Miksch et al.[22] 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.[4] used a sampling technique to abstract time-series data and introduced DOI (degree of interest) functions to determine the sampling rate. Konyha et al. described a visual analysis framework to help data analysts explore large-scale time-series data containing hundreds of variables .[23] In order to use an abstraction algorithm, a distorted timeline might be necessary to give important data more space. This technique is used in many research efforts .[3–6,24]

Some visualization techniques and systems have been designed and implemented for particular types of time-series data. Some researchers focus on univariate data. Hao et al. used variable resolution density displays to visualize univariate data.[25] They designed circular overlay displays to avoid data shift movements after the display is full. *BinX*[26] 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.

Several research efforts involve the visualization of multidimensional correlations. Wong et al.[27] present techniques for handling a multidimensional data stream. They focused on how to reduce the time complexity for generating scatterplots for visually conveying clusters in data streams. Hao et al. used an importance-driven layout to represent the degree of importance for different dimensions via assigning the important dimensions more space.[28] The Intelligent Visual Analytics Query (IVQuery) is a visual analysis tool to help users perceive relationships among multiple time-series data dimensions.[29] *TimeWheel* puts the time axis in the center and other data dimensions arranged circularly. Each data item corresponds to a group of lines from the time axis to the axes for data dimensions .[30] Yu et al. developed a tool for the visual analysis of multi-stream multimedia data.[31] Users can highlight selected data portions, or zoom in on the region of interest to study the data trends and the multivariate data patterns. Ward and Guo employed N-gram approach to map temporal data to multiple glyphs, and then utilize the PCA algorithm to position these glyphs. Thus various patterns can be recognized.[32] Compared to the above efforts, we focus on how to visualize changes of data patterns in time-series data.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed a set of history views to help users explore data pattern changes within a relatively long time period. Users can define a hierarchical structure to represent timelines. The definition of the hierarchy can be from natural time units, such as year, quarter, and month, or domain specific units. For a specific time range selected by a user, we can generate the history view containing all time windows in this time duration. In the non-merged mode, the abstraction of

data patterns is drawn in glyphs instead of the original datapoints. All glyphs comprise a grid or virtual calendar. Other approaches, such as an MDS pattern starfield, and distance map, are designed to assist users in studying the similarities of time windows. Under the merge mode, time windows selected by users are sent to the merge algorithm and a merge-based hierarchy is generated. Users can choose an appropriate level to study the pattern changes. The usability evaluation demonstrated that most users could understand the concepts in the history views and finish the assigned tasks, including navigating timelines, finding significant pattern changes, and investigating similarity among time windows. Some future works include:

- We plan to apply the proposed approaches to a variety of real domains, such as financial data analysis and video monitoring. It may be necessary to modify the proposed framework and design more visualization and interaction techniques for these applications.
- To improve the validation of our visualization techniques, we plan to work with domain experts to assess the implemented system using dataset from their areas.

ACKNOWLEDGMENTS

This work is supported under NSF grant CCF-0811510.

REFERENCES

[1] Aigner, W., Miksch, S., Müller, W., Schumann, H., and Tominski, C., "Visual methods for analyzing time-oriented data," *IEEE Transactions on Visualization and Computer Graphics* **14**(1), 47–60 (2008).

[2] Aigner, W., Miksch, S., Schumann, H., and Tominski, C., [*Visualization of Time-Oriented Data*], Springer, New York, NY, USA (2011).

[3] Bade, R., Schlechtweg, S., and Miksch, S., "Connecting time-oriented data and information to a coherent interactive visualization," *CHI* , 105–112 (2004).

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

[5] Miksch, S., Seyfang, A., Horn, W., and Popow, C., "Abstracting steady qualitative descriptions over time from noisy, high-frequency data," *Proc. Artificial Intelligence in Medicine. Joint European Conference on Artificial Intelligence in Medicine and Medical Decision Making* , 281–290 (1999).

[6] Wattenberg, M., "Baby names, visualization, and social data analysis," *Proc. IEEE Symp. Information Visualization* , 1–7 (2005).

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

[8] Xie, Z., Ward, M. O., and Rundensteiner, E. A., "Visual exploration of stream pattern changes using a data-driven framework," *Proc. 6th International Symposium on Visual Computing* **6454**, 522–532 (2010).

[9] Wijk, J. V. and Selow, E. V., "Cluster and calendar based visualization of time series data," *Proc. IEEE Symposium on Information Visualization* , 4–9 (1999).

[10] Meyer, M. D., Munzner, T., DePace, A. H., and Pfister, H., "Multeesum: A tool for comparative spatial and temporal gene expression data," *IEEE Trans. Vis. Comput. Graph.* **16**(6), 908–917 (2010).

[11] Kruskal, J. and Wish, M., [*Multidimensional Scaling*], Sage Publications, Thousand Oaks, CA, USA (1978).

[12] Buja, A., Swayne, D. F., Littman, M. L., Dean, N., and Heike Hofmann, L. C., "Data visualization with multidimensional scaling," *Journal of Computational and Graphical Statistics* **17**(2), 444–472 (2008).

[13] Yang, J., Hubball, D., Ward, M., Rundensteiner, E., and Ribarsky, W., "Value and relation display: Interactive visual exploration of large data sets with hundreds of dimensions," *IEEE Trans. Visualization and Computer Graphics* **13**(3), 494–507 (2007).

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

[15] Ward, M. O., Grinstein, G., and Keim, D., [*Interactive Data Visualization: Foundations, Techniques, and Applications*], A K Peters Ltd, Natick, MA, USA (2010).

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

[17] Tufte, E., [*The Visual Display of Quantitative Information*], Graphics Press, Cheshire, CT, USA (1983).

[18] Carlis, J. V. and Konstan, J. A., "Interactive visualization of serial periodic data," *Proc. Symp. User Interface Software and Technology* , 29–38 (1998).

[19] Hewagamage, K. P., Hirakawa, M., and Ichikawa, T., "Interactive visualization of spatiotemporal patterns using spirals on a geographical map," *Proc. Symp. Visual languages* , 296–303 (1999).

[20] Ward, M. and Lipchak, B., "A visualization tool for exploratory analysis of cyclic multivariate data," *Metrika* **51**(1), 27–37 (2000).

[21] Weber, M., Alexa, M., and Müller, W., "Visualizing time-series on spirals," *Proc. IEEE Symp. Information Visualization* , 7–14 (2001).

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

[23] Konyha, Z., Matković, K., Gracanin, D., Jelovic, M., and Hauser, H., "Interactive visual analysis of families of function graphs," *IEEE Transactions on Visualization and Computer Graphics* **12**(6), 1373–1385 (2006).

[24] Huynh, D. F., "SIMILE – timeline [http://simile.mit.edu/timeline/]," *Massachusetts Institute of Technology* (2006).

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

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

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

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

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

[30] Tominski, C., Abello, J., and Schumann, H., "Axes-based visualizations with radial layouts," *Proc. ACM Symp.on Applied Computing* , 1242–1247 (2004).

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

[32] Ward, M. O. and Guo, Z., "Visual exploration of time-series data with shape space projections," *Computer Graphics Forum* **30**(3), 701–710 (2011).