Stack Zooming for Multi-Focus Interaction in Time-Series Data Visualization

Waqas Javed and Niklas Elmqvist, *Member, IEEE*

Abstract—Information visualization shows tremendous potential for helping both expert and casual users alike make sense of temporal data, but current time series visualization tools provide poor support for comparing several foci in a temporal dataset while retaining context and distance awareness. We introduce a method for supporting this kind of multi-focus interaction that we call *stack zooming*. The approach is based on interactively building hierarchies of 1D strips stacked on top of each other, where each subsequent stack represents a higher zoom level, and sibling strips represent branches in the visual exploration. These hierarchies can also be used as graphical histories and for communicating insights. Correlation graphics show the relation between stacks and strips of different levels, providing context and distance awareness. We also discuss how visual spaces that support stack zooming can be extended with annotation and local statistics computations that fit the hierarchical stacking metaphor.

Index Terms—Temporal data, multi-focus interaction, comparative visualization, visual exploration, visual analytics, interaction.

1 Introduction

Almost all data found in the real world have a temporal component, or can be visualized with respect to time. Temporal data is prevalent in virtually all scientific domains, and permeates societal phenomena as well—examples include computer and network logs, chronological history, political poll data, stock market information, and climate measurements. One of the common properties of this wide range of data is that the datasets tend to be large, not only in the captured time period but also in the number of observed attributes [2]. Visualization has been suggested as a way to handle this problem (e.g., [16]), but effectively exploring large-scale temporal datasets of this kind often requires *multi-focus interaction* [10], i.e. the capability to simultaneously view several focus regions of a dataset at a sufficiently high detail while retaining context and distance awareness. Consider for example a stock market analyst who is trying to predict future market trends using standard line graphs of stock market indices by extrapolating from several similar situations at different times in the past.

Most existing temporal visualization tools do not fully support the requirements of multi-focus interaction, including TimeSearcher[16], ATLAS [9], PatternFinder [12], LifeLines [22, 36], and Growing Polygons [11]. Major obstacles in completely supporting multi-focus interaction include space management for the guaranteed visibility of focus points, context and interval awareness between two focus points, and visibility of a synchronous view for the user while panning the focus points through the datasets. For our stock market analyst, this translates to seeing several time periods of the stock market graphs simultaneously while understanding their individual relations, temporal distances, and the market developments before and after each period.

In this paper, we present a general technique to supporting multi-focus interaction for one-dimensional visual spaces—which includes the temporal datasets motivating this work—that we call *stack zooming*. *Stack zooming* is based on a user study, conducted to analyze, how humans comprehend the time series data while focusing on different regions of a very large scale time series. In this user study, a contextual inquiry method [**?**] is used to collect information related to the steps a user follows, if asked to compare different portions of the time series and to predict any future trends. Data collected through this study is used to develop *stack zooming* technique. To exemplify the new method, we also present a prototype implementation, called TRAXPLORER (Figure 1), that supports multi-focus interaction in temporal data through stack zooming, as well as additional functionality such as local statistics computation and annotation support. Our prototype could potentially help our financial analyst by providing an integrated view of the whole as well as selected parts of the stock market timeline and allow for easy and direct exploration of the data.

Beyond directly supporting visual analysis, the stack hierarchy also serves as a tangible graphical history [7, 15, 29] of the visual exploration session. Recent advances in visual communication have suggested that to ameliorate the process of sensemaking [25], visualization should also support collaboration and communication [33, 35]. Our TraXplorer prototype takes this a step further by introducing a *presentation tree* that is automatically built during exploration, and can then double as a management interface for annotations, computations, and, ultimately, presentation of the results of the exploration session. To complete our stock market analysis example, this functionality would enable our analyst to present his or her predictions of future market trends to a manager in a step-by-step fashion, progressively expanding selected parts of the annotated exploration history and displaying relevant statistics integrated with the visual display.

The contributions of this paper are the following: *(i)* the general stack zooming technique and its implications to visualization of time-series data; *(ii)* the implementation of our prototype system for temporal visualization based on stack zooming and with support for communication through automatic exploration history capture; and *(iii)* a usage scenario conducted with the TraXplorer system showing how it can be used for visual exploration of a large financial time-series dataset.

In the next section we discuss the related work on time-series visualization, multi-focus exploration, and collaborative information visualization. We then present the general stack zooming model to create the track hierarchy. Next, we discuss the TraXplorer system, followed by implementation notes for the framework and a case study of the usage scenario for stock market data. We conclude the paper by discussing some of the design decisions and weaknesses of stack zooming, and round off with our conclusions and plans for future work.

2 Background

In the following subsections, we give the background of our time-series visualization, starting with the state-of-the-art in temporal visualization, and then progressing into support for focus+context interaction [13] that few existing tools fully support. We also discuss aspects of collaboration and communication using visualization that integrate nicely with these ideas.

- *Waqas Javed is with Purdue University in West Lafayette, USA, E-mail: wjaved@purdue.edu.*
- *Niklas Elmqvist is with Purdue University in West Lafayette, USA, E-mail: elm@purdue.edu.*

Manuscript received 31 March 2007; accepted 1 August 2007; posted online 27 October 2007.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

2.1 Time-Series Visualization

In the last two decades, a large number of research papers have been published related to visualization of time-series datasets—relevant surveys include [20, 30]. During this time, temporal visualization has evolved from basic timeline graphs [34] (some of these dating back to the tenth century), through becoming a novel approach for using visualization as a problem solving technique [7], and finally to encompass design considerations of sophisticated systems like ATLAS [9] for visualizing massive time-series datasets.

Correspondingly, considerable research efforts have been directed towards developing methods for interactive visual exploration of time-series data; here follows a representative sampling. The Perspective Wall [18] presents one-dimensional temporal data using a 3D rendering that incorporates a natural focus+context [13] perspective distortion. LifeLines [22] was one of the early systems that uses visualization to explore personal histories (Wang et al. [36] recently presented a follow-up to the original LifeLines system). TimeSearcher [16] is a time-series visualization tool that uses timeboxes to generate visual queries to explore the datasets. PatternFinder [12] provides an interface for querying temporal data of medical records. Continuum [4] is a Web 2.0 tool for faceted timeline browsing. Most recently, LiveRAC [19] is an interactive visual exploration environment for time-series data in system management.

All of these projects concentrate mainly on the interactive exploration of the time-series datasets. However, beyond standard low-level visualization tasks [3], visual exploration in temporal datasets often require additional attention to the special properties of the time dimension [1, 2]. In particular, additional important temporal analysis tasks include [4] support for (a) dynamic temporal hierarchies, (b) across-concept relationships, and (c) large-scale overviews. Few of the above time visualization tools, with the exception of LiveRAC [19] and Continuum [4], explicitly support all of these requirements.

2.2 Supporting Multi-Focus Interaction

We argue that these extended temporal analysis tasks can be generalized to the concept of multi-focus interaction [10], a conceptual framework that integrates multiple focus+context [13] views with guaranteed visibility to support both focus, context, and distance awareness. This is the approach taken by the LiveRAC system, implemented using the Accordion Drawing framework [21, 31] for guaranteed focus visibility. In this paper, we take an alternative approach to multi-focus interaction than LiveRAC, taking advantage of the one-dimensionality of the data to present hierarchies of undistorted visualization strips, as opposed to a single, integrated—but visually distorted—view.

A number of general techniques have evolved to support multi-focus interaction, e.g., split-screen [28], fisheye views [13], and space-folding [10], etc. However, none of these techniques was originally developed for the exploration of time-series datasets. The Continuum faceted timeline browser [4] implements the above extended temporal tasks, but provides only one level of overview, meaning that detail and context awareness may be limited for a complex zoom stack.

Two existing systems are of particular relevance to the stack zooming technique presented in this work. The multi-resolution time slider presented by Richter et al. [23] uses a hierarchical zoom stack like we do, but is primarily an interaction technique for selecting time periods in a hierarchical fashion and does not appear to support multiple focus points. The idea has also not been fully developed for visual exploration. Second, the multi-page zooming technique presented by Robert and Lecolinet [24] defines a hierarchical zooming technique similar to our stack zooming, but is defined for hypertext document graphs and has a different presentation approach compared to us.

2.3 Supporting Communication and Collaboration

Advances in information visualization research have showcased the power of collaboration to amplify sensemaking [25] and enable distributed cognition [17] in a team of analysts, but collaboration requires a strong communication component to disseminate insights across participants [33, 35]. Of course, communication-minded visualization is also important in its own right for ultimately disseminating the results of visual exploration to stakeholders outside the analysis team.

A recent approach in general human-computer interaction has been to utilize automatically captured interaction histories as a means of fostering awareness of your own (e.g. [8]) and your team's (e.g. [6]) activities. Self and group awareness is a big research topic that is largely outside the scope of this article—we concern ourselves with the specific use of visual exploration histories to support communication.

Branching interaction histories for visualization that capture multiple exploration paths were introduced as early as in the GRASPARC [7] system, and was recently utilized to great effect by Shrinivasan and van Wijk [29] for a scatterplot visualization. Their tool consists of a data view, a navigation view, and a knowledge view. However, their history mechanism stores all interactions in a visual exploration system, which can be a large and complex dataset with no real connection to the useful outcome from the analysis. In contrast, Heer et al. [15] use linear (non-branching) graphical histories in the Tableau (formerly Polaris [32]) visualization framework. Their concept of event compression is interesting, but lacks a way to harness the interaction history for authoring presentations to communicate insights to stakeholders.

3 Stack Zooming

Consider a user trying to analyze a large-scale temporal dataset using a line graph such as the one in Figure 3. Because a large-scale time-series dataset often contains more data points than can be fit on the screen, the user will need to zoom and pan in the line graph to see details necessary to solve a particular task. However, using just a sequence of zoom and pan actions to support higher-level temporal analysis tasks such as comparison and correlation [2, 4], both between different time-series and within the same time-series, will quickly become tedious and ineffective [13, 14].

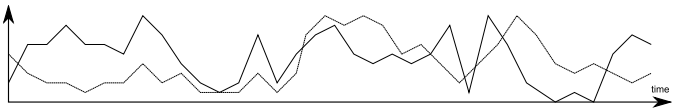
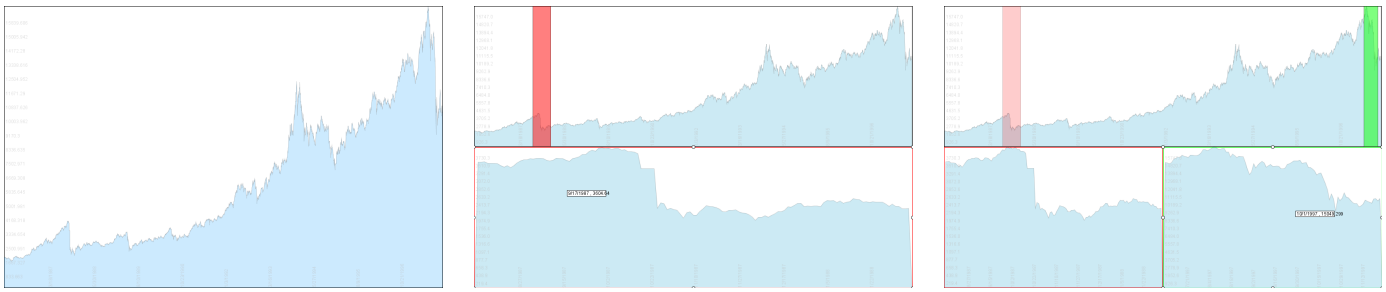


Fig. 2. Two time-series datasets visualized as line graphs.

With the *stack zooming* technique, the available display space is split evenly among the currently visible line charts, or *strips*, that show subsets of the time-series dataset. When the user begins to analyze the dataset, the whole display is taken up by the full time-series drawn as a line visualization on a single strip. Using a mouse drag on the surface of this strip, the user can create a child strip of the main strip that displays the selected subset of the time data. Additional zoom operations on the main dataset strip will create additional children in the *zoom stack*, all of them allocated an equal amount of the available display space for that particular level (space allocations can be changed by dragging the borders of a strip). Each child strip serves as a focus point, and is guaranteed to always be visible on the visual space.

Color-coded frames for the child strips and correspondingly color-coded selection areas in the parent strips show the correlation between parents and children, as well as provide intervening context and distance awareness between the focus points. Dragging a selection area in a parent strip will pan the connected child strip, and children can be panned directly by using the arrow keys. In this way, a user can quickly explore the temporal dataset using a sequence of simple zoom and pans while retaining the multi-focus interaction support.



(a) (b) (c)

Fig. 1. Stack zooming in the Hongkong stock market index dataset. (a) Main strip showing the whole dataset (June 1986 to December 1997). (b) Creating a child strip for Black Monday (October 19, 1987). (c) Creating a second child strip for the Asian financial crisis (starting July 1997). Transitions between one view are animated, and the layout algorithm ensures that the whole space is used.

In the example in Figure 2(a), the user is studying a financial dataset of the Hong Kong stock exchange (HNGKNGI) from June 1986 to December 1997. He first creates a child strip for the Black Monday crash of October 19, 1987 by dragging on the main strip (see Figure 2(b)). Then, interested in comparing this crash to the Asian financial crisis in 1997, he creates a second child strip centered around the period July to October, 1997 (Figure 2(c)). A continued description of this usage scenario can be found in Section 5 of this paper.

The remainder of this section will introduce the theoretical structure and operations of stack zooming. We then describe the layout mechanism and how to handle focus point merging. We end the description by discussing interaction techniques for controlling the zoom stack.

3.1 Zoom Stacks

The basic structure in stack zooming is a hierarchical *zoom stack* (or *Z-stack*)[1]. The zoom stack describes the layout of the hierarchy of focus regions, orthogonal to the underlying one-dimensional dataset the tree operates on (i.e. the zoom stack can manage any one-dimensional visual space). Just like any other tree, a Z-stack is defined by a single root node r containing all of the nodes of the tree. More formally,

$$\textit{Z-stack} = r.$$

The Z-stack structure defines all basic tree operations such as finding the depth of individual nodes or the whole tree, the number of children, and all standard tree traversals.

Nodes in a zoom stack are called *zoom nodes* (*Z-nodes*). A single Z-node n captures a single *strip* (a focus region in the 1D visual space) in the stack zooming technique. Thus, the node n consists of a range $[t_0, t_1]$ describing the extents of the focus region in data dimension space, a $disp-dim$ factor describing the layout allocation for this particular node on the screen, a parent node p, and an ordered list of child nodes C. In formal terms,

$$\textit{Z-node} = (t_0, t_1, \textit{alloc}, p, C).$$

Screen allocations (*alloc*) are specified as scale-independent ratios of the full allocation of the whole zoom stack. This measure, along with the node order in the list of children for the parent, governs the actual screen location of the node when it is visualized.

Zoom nodes explicitly capture a focus region on the data, which is why it is specified in terms of an interval (as opposed to a specific point and a scale factor). This is so that giving more screen space to a visualization of a particular Z-node (i.e. increasing the node's *alloc*) will cause the data in the fixed interval $[t_0, t_1]$ to be magnified to allow for easier inspection, as opposed to exposing more data.

[1]Note that zoom stacks are trees and not linear lists, like normal stacks.

3.2 Layout

Zoom nodes in a zoom stack are laid out on the visual 2D substrate using a space-filling layout algorithm (Figure 4) that splits the vertical space by the height of the tree (assigning an equal amount to each tree level), and the horizontal space by the number of siblings on a level basis for each level of the tree (assigning an equal amount to each sibling). At any point in time, the user is able to change the layout allocations by dragging the borders of individual strips.

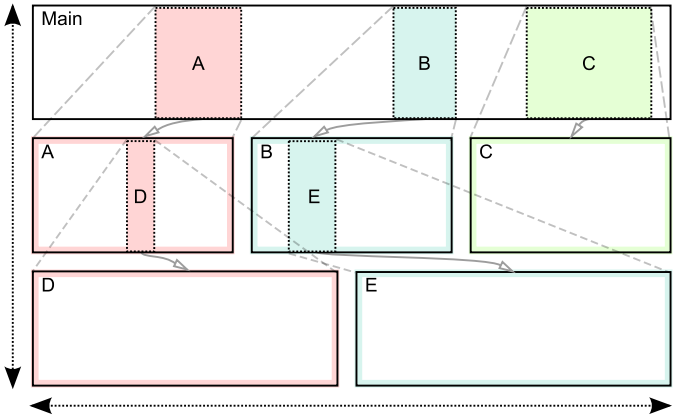


Fig. 3. General layout mechanism for the stack zooming technique. Color-coded zoom areas and corresponding colored frames show parent-child relationships, and visual arrows make the relations explicit.

The ordering of child strips for each level may be significant for the purpose of conveying the relative positions of the displayed intervals of a time series to the user. Therefore, the layout manager will always order child strips for each level in the zoom stack to be the same as the order of their intervals on the parent strip. Overlapping intervals, such as when moving focus points, is a special case—see Section 3.4.

To preserve the tree structure of the zoom stack, one design alternative is to not divide space equally across siblings of each level of the zoom stack, but rather to manage space on a per-subtree level. This would give an explicit indication of the parent-child relationships between strips in adjacent levels, and thus decrease the need for explicit correlation graphics (discussed next). However, we found that because visual exploration using stack zooming often results in unbalanced zoom trees, this would result in suboptimal use of the available screen space. Therefore, global space allocation across each level is a better solution.

3.3 Correlation Graphics

If we are to retain focus, context, and distance awareness for a visual space supporting stack zooming, we need to make explicit the rela-

tionship between parent strips and child strips in adjacent levels of the zoom stack. For the reasons outlined above, we cannot directly show ancestor relationships in the layout, or we may waste valuable screen space. Therefore, we introduce a set of *correlation graphics* that visually indicate the relationships between parents and children.

The correlation graphics take several forms (see Figure 4):

- **Color-coded zoom areas:** Parent strips show color-coded (but semi-transparent) selection areas that indicate the position and extents of each child strip in the time-series. These areas can also be interacted with, see the next section.
- **Color-coded strip frames:** Child strips have color-coded frames that correspond to the color of its selection area in the parent. This gives a visual link between parent and child.
- **Correlation links:** For a complex visual exploration session, the above two graphics may not be sufficient to quickly understand the correlation structure in the zoom stack. We introduce explicit correlation links drawn as arrows from zoom areas in parents to the respective children. In an effort to minimize visual clutter, an implementation may choose to expose these arrows as a transient overlay (e.g., shown only when the user is holding down the Control key or have toggled a checkbox, for example).

3.4 Navigation

One of the basic requirements of effective exploration of a time-series datasets is to allow the user to navigate the focus points through the dataset. We have already described the zoom functionality, which creates new zoom strips in the stack and clearly is intrinsic to stack zooming. The other main navigation operation is *panning*—or moving—a focus point. In our technique, moving a focus point can either be done by dragging the zoom area selections in a parent strip, or by panning a child strip directly, such as with the keyboard arrows keys, or by a dragging mouse gesture at the left and right border of the strip.

Moving a child strip may give rise to a special layout case when the interval covered by one strip overlaps that of another strip. Recall that our layout mechanism always attempts to preserve the spatial order of strips for each level according to their temporal order in the parent strips. During overlap, the layout will not be changed to maintain stability of the display, but if the temporal order of two strips change as a result of a navigation operation (i.e. a pan), the layout will switch the relative position of the two affected strips on the visual space.

Overlapping has special potential for reinforcing the awareness of a user navigating in the time series by merging adjacent child strips when their intervals overlap. Figure 5 gives a schematic overview in both layout and temporal space of this operation. However, strip merging requires that all zoom strips in a stack level cover the same length of interval W (i.e., same zoom factor), or visual stability will not be maintained. Therefore, it may not be practical for all applications.

4 The TraXplorer System

The TraXplorer system is a time-series visualization tool supporting multi-focus interaction using the stack zooming technique introduced in this paper. Time series are represented as *tracks*, hence the name of the tool. The system was designed to support a communication-minded [35] iterative workflow process depicted in Figure 6 where there are three phases involved: individual exploration, collaboration within the analysis team, and dissemination to external stakeholders. For the TraXplorer tool, this is realized in the following ways (explained in subsequent sections):

- **Exploration:** An individual analyst, or potentially a number of analysts, use the stack zooming technique to explore complex time-series dataset in the tool and gain insights about the data (Section 4.1). Here, the zoom stack doubles as a branching exploration history graph [7, 29].
- **Collaboration:** The analyst uses the zoom stack as an exploration history to communicate progress and insights to other members of the analysis team (Section 4.2). The analysis process can now be continued, potentially by other analysts, by backtracking to the exploration phase. We provide a method for pruning, annotating, and calculating local statistics for individual strips or strip subhierarchies in the exploration history.
- **Dissemination:** Insights gained during the internal exploration and collaboration phases are presented to external stakeholders (Section 4.3). The TraXplorer tool has a dissemination mode using the standard visual representations, but the branching exploration history is linearized into a sequence (akin to a slide show) to better support communicating insights to non-experts.

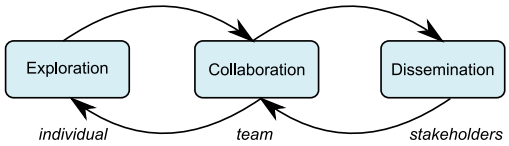


Fig. 5. Communication-minded visualization workflow for TraXplorer.

The above workflow process gives a clear path from exploration to dissemination [33], and bridges the current gap between visualization tools and traditional presentation tools such as Microsoft PowerPoint.

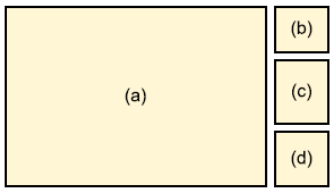


Fig. 6. The TraXplorer interface (screenshot in Figure 1). (a) Main visualization window. (b) Presentation tree. (c) Data box. (d) Layer control.

4.1 Visual Exploration

The TraXplorer exploration interface consists of the main visualization window (Figure 7(a)), the data box for drilling down into the dataset (Figure 7(c)), and the layer control box (Figure 7(d)) for managing the visual display of the time series data. The main functionality of the tool is to support loading several tracks of time-series data (defined as comma-separated files, or similar) into the same visual space.

The main visualization window is a visual space supporting stack zooming. It contains a set of visualizations of time-series data on a common time axis and potentially different value axes. The type of visualization is independent of the layout management—our implementation currently supports color-coded line graphs. Multiple time series will be drawn as tracks overlaid on the same temporal axis. To help manage these tracks, the layer control box can be used to move, to delete, and to toggle the visibility of individual tracks, as well as change color mapping, transparency, and track title. Furthermore, using the layer control, two or several tracks can be *linked* to use the same value (Y) axis to allow for pairwise comparison. If tracks are not linked, they will be scaled to use the whole vertical space of each visualization strip, independent of other tracks. Layer control is also used to determine which track should be used for the value axis labels.

Our implementation of the stack zooming technique for the main visualization window allows for easily exploring the time series data by building zoom stacks. More specifically, dragging on a time interval on a visualization strip will immediately create a child strip for that

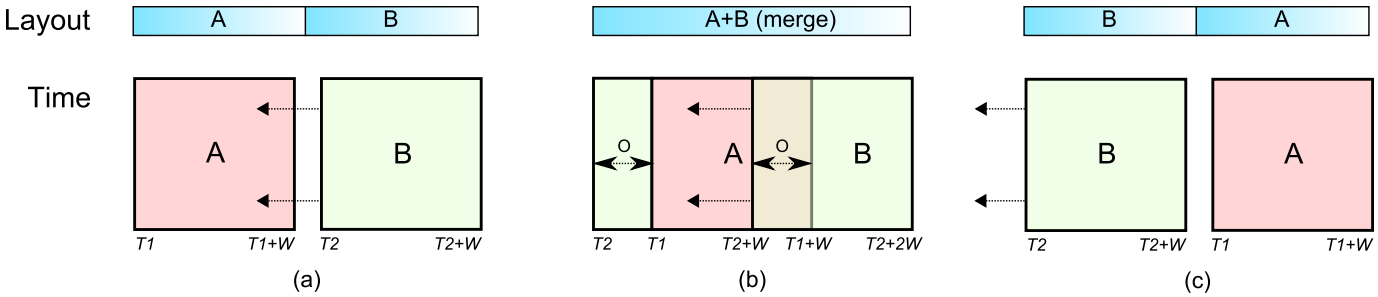


Fig. 4. Layout and temporal view of merge operations for strips during temporal overlap. (a) Two strips, A and B, at time positions $T1$ and $T2$, respectively, and both of interval width W are approaching. (b) Temporal overlap between A and B, causing the layouts to merge into a single strip. To maintain the same combined layout size on the visual space, all temporal overlap is added to the beginning of the combined interval. (c) B has now passed A and there is no longer a temporal overlap, so the strips are separate and with correct temporal order in layout space.

interval. Color-coded correlation graphics will show the relationship. Dragging the time window graphic in a parent will pan the corresponding child. Intervals are positioned relative to the parent's interval, so dragging a strip will also pan its children the same amount. Each strip has additional decorations to allow for maximizing the size of a specific strip on the layout space, hiding it (see below), as well as deleting it. Deleting a strip will delete all of its children. Furthermore, dragging on the border of a strip enables direct resizing to fit user taste.

To further support visual exploration, we also provide a data box that gives local statistics about the currently selected strip in the zoom stack. This provides detail-on-demand functionality for computing measures such as minimum, maximum, average, median, and standard deviation metrics for a particular track. The data box also doubles as a presentation tool; this is discussed more closely in the next section. In addition, users can always hover their mouse over a particular time series point to retrieve exact data about the point in a tooltip.

4.2 Collaboration

The TraXplorer objective is to support an analyst communicating insights gained during an individual exploration session to the whole team, while allowing for continuing the exploration in collaborative mode, or iterating back to individual exploration (by the same analyst or another one) after conferring with other team members. In other words, for this setting, the analyst will need support from the visualization tool for a semi-structured presentation, while at the same time being able to explore different branches and create entirely new ones.

The zoom stack already captures the exploration history, and we reinforce this by providing a presentation tree (Figure 7(b)) that maintains a hierarchical representation of the zoom stack. This approach is similar to the branching exploration histories of GRASPARC [7] as well as of Shrinivasan and van Wijk [29].

In our work, we take the history concept a step further by utilizing the history as a starting point for creating a presentation of the exploration. The analyst can prune, move, or hide individual nodes (i.e. child strips) in the presentation tree to refine the presentation, while at the same time being able to extend the presentation by stack zooming in the visualization at any time, even during the actual presentation. This particular feature is important to allow seamless transitions between exploration and presentation. Also, by selectively hiding and showing parts of the zoom stack using the presentation tree interface, analysts can better structure their narrative while presenting their insights.

4.3 External Dissemination

Finally, there comes a point when the insights gained by an analyst team must be communicated to the external, and often non-expert, stakeholders [35]. Our aim with TraXplorer was to bridge the gap between the visualization tool and the presentation tool, eliminating the need to, for example, cut and paste visualization images into traditional presentation applications like Microsoft PowerPoint.

Instead, the analyst (or the team of analysts) can use the exploration history contained in the presentation tree to *linearize* the combined exploration sessions of the data, i.e., to create a linear sequence of views similar to a slideshow presentation suitable for presentation to the audience. Because stakeholders may not be experts in data analysis, this functionality allows for hiding some of the complexity of the exploration tool and just show familiar line graphs in fullscreen mode. The analyst can still revert back to the whole branching exploration history, even during presentation time, if needed.

Mere images may not be enough for an effective presentation, so the data box that supports details-on-demand in the exploration phase also doubles as an annotation mechanism. Checkboxes in the data box for each computed and user-supplied metric (i.e. annotation) allows the user to also display these metrics on the actual visualization strip. In other words, this functionality can add local statistics—such as a key, the extrema, or the average of a particular data track—to the actual visual display of the track. The order of displayed metrics can also be captured in the linear presentation sequence, allowing for these metrics to be progressively displayed in the slideshow presentation.

4.4 Implementation Notes

The TRAXPLORER system was implemented in Java using the Piccolo[2] structured 2D graphics toolkit [5]. It accepts basic comma-separated (.csv) data files as input, and enables the user to select any two dimensions in the dataset as the time (X) and value (Y) axes for a particular time-series. This way, the user can easily load several data series into the tool to allow for comparison between different datasets and not just within the same dataset.

The key components in the prototype implementation include the time strip class (implemented as subclasses of Piccolo's `PNode` basic scene graph node class), the actual visualization (which is orthogonal to the strips and the layout), and the layout manager responsible for calculating and animating the layouts for a hierarchy of strip nodes. Although a recursive layout algorithm may seem to fit the zoom stack hierarchy, our implementation does not use recursion because the layout on each level is actually performed globally for the whole level, and not for the particular space allocation of the parent.

Another implementation consideration is the design of the presentation interface. Our prototype makes no distinction between exploration and presentation mode, but this might be a suitable distinction to make. In particular, one could envision a multi-display setup, where the primary display shows the visualization for the audience using an LCD projector, and the secondary display is on the computer screen and shows

[2]http://www.piccolo2d.org

the presentation interface. This way, the display shown to the audience will not be burdened by the complexities of the exploration and presentation interfaces, but will be solely devoted to the visualization.

5 Usage Scenario

To show how TraXplorer could be utilized to analyze a particular dataset, we return to our stock market analysis example discussed earlier. Our analyst, Joe, is part of a team of financial analysts studying the history of stock market crashes to predict future behavior.

Joe has been tasked to study two crashes in particular: the Black Monday crash of October 1987, and the Asian financial crisis of July 1997. Because these events are spaced ten years apart in a large temporal dataset (daily values for all major stock market indices), Joe knows he will need multiple focus points, and thus loads the datasets into TraXplorer. Since the crashes affected European and Asian markets differently, he decides to study one representative index for each continent, and loads the UK (FTSE100) and Hong Kong (HNGKNGI) time series from June 1986 to December 1997. Looking at the initial view of these two tracks (Figure 8), Joe notes that both indices were clearly affected by the two crashes. Pairwise comparison does not make sense for this task, because the two indices have different scales, so he does not link the tracks, but he uses the exploration interface to change transparency and toggle their visibility to better see the individual values.

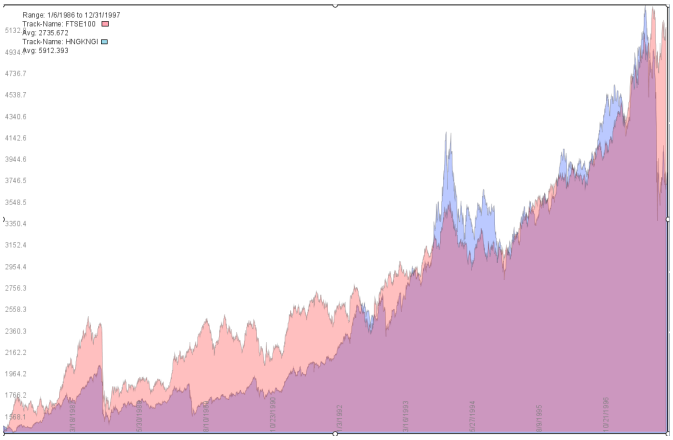


Fig. 7. UK (FTSE100, pink) and Hong Kong (HNKNGI, blue/purple) values for the period October 1987 to December 1997.

Interested in understanding the buildup and aftermath of a crash, Joe creates two focus points—both spanning over an interval of two years—around the events to explore the behavior of the market before and after the crashes (Figure 9). He notes that both indices have a steeply increasing trend prior to a crash, and the HNGKNGI index appears more effected during the Asian market crisis of 1997. From this data, Joe hypothesizes that there was a "bubble" effect in both cases, with the 1997 bubble being more confined to the Asian markets.

Next, Joe decides to seek more insight into market behavior to predict future market crashes. Looking at the two focus points, he creates two additional time windows in the Black Monday focus point. He centers one of them at the crash date and navigates the other time window to approximately a year before the crash time. Joe also resizes the child strips to give more visual space to the new focus points and enables local statistics in them (Figure 10). By looking at these views, he notes that within only a year before the crash, both FTSE100 and HNGKNGI sees a steep increase of approximately 55% and 100%, respectively, and once the crash occurred there was a 35% drop in FTSE100 and a 50% drop in HNGKNGI index.

Impressed by these results, he decides to repeat the same procedure for the Asian financial crisis of 1997 (Figure 11). Here, he observes

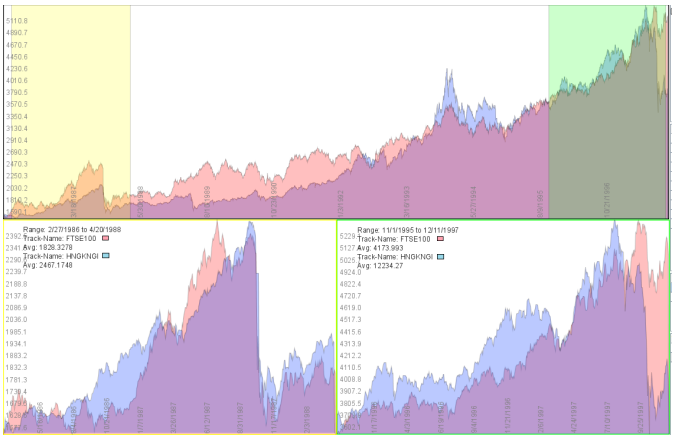


Fig. 8. Details for 1987 and 1997 crashes. (UK pink, Hong Kong blue.)

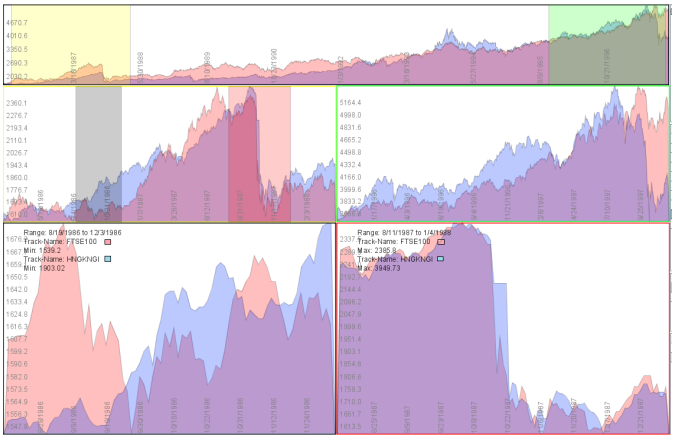


Fig. 9. Exploring buildup and aftermath of the Black Monday crash.

that this time around, there are approximately 50% and 60% increases in the FTSE100 and HNGKNGI indices, respectively, within a year before the crash. He also observes that within a week after the crash, there is a 12% drop in FTSE100 and a 46% drop in HNGKNGI. Based on these observations, he infers that a very steep "boom" (more than 50% increase) in the market index over a period of more than one year can signify a "bubble" and raises the chances of a major market crash. He also observes that in comparison to Black Monday, the UK market was less effected than the Hong Kong market by the 1997 crash.

Once done with his exploration of the two datasets, Joe decides to share his findings with his colleagues. To add more force to his points, he uses the presentation interface of the TraXplorer system to explain his exploration process. In doing so, Joe can use the system itself to prune away unused exploration branches and streamline his presentation. In addition, as he is communicating his findins, he is able to return to the exploration phase at any time as the discussion among the team members progresses. During the discussion, the team observes that markets remain unstable for next few months following any major crash. After their discussion, the team decides on a linear presentation sequence of their results for an upcoming board meeting.

6 Discussion

It is worth pointing out that although this paper has focused on visualizing temporal data, the stack zooming technique is really not limited to such datasets, but can be utilized for any one-dimensional—or virtually one-dimensional—visual space, which is what a timeline representation of a time-series dataset usually is. "Virtually one-

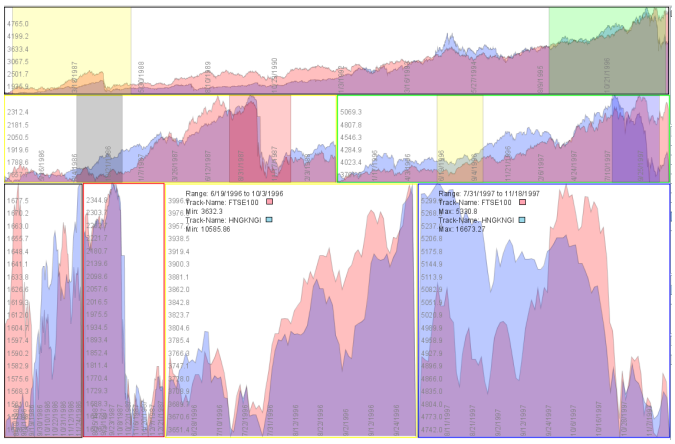


Fig. 10. Exploring buildup and aftermath of the 1997 Asian crash.

dimensional" means that the technique could be applied to any other dataset that has a natural extension in one dimension, and limited in another. Examples of this kind of dataset includes multimedia data such as video (the multi-scale slider presented by Richter at al. [23] uses a similar idea to ours for this very purpose), text axnd hypertext documents (see the multi-page zooming technique by Robert and Lecolinet [24]), space-time (Hasse) diagrams (such as the causality data visualized by Growing Polygons [11]), and bipartite graphs.

The reason for this restriction is obviously that stack zooming uses one of the dimensions of the visual space for layout. It may be argued that we could support stack zooming in an inherently two-dimensional visual space, such as a treemap [27], if we only extended our technique to three dimensions, and thus used the Z (depth) axis for layout. However, this is outside the scope of this paper, and left for future work.

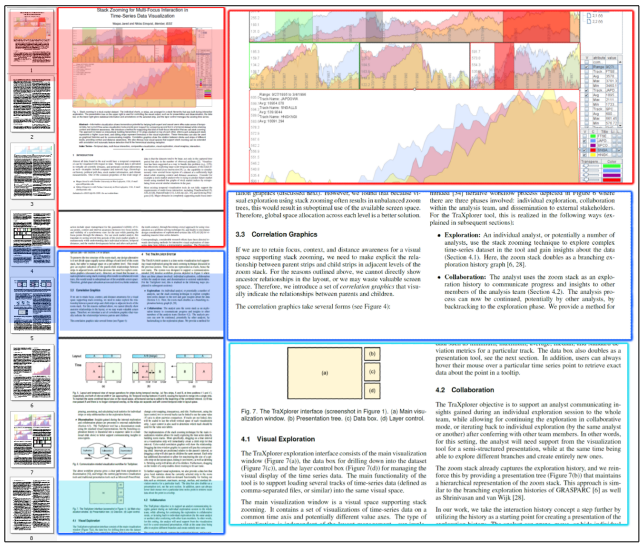


Fig. 11. Stack zooming in vertical layout for a multi-scale document (mockup screenshot, not implemented in this paper).

To illustrate another use of stack zooming, Figure 12 shows an example of a multi-scale representation of a document visualized using stack zooming. This approach might be useful for comparing several passages of a long or complex document (e.g., when studying an advanced software manual), or for showing several occurrences of a specific word or phrase as well as their surrounding context.

We designed the stack zooming technique to be an alternative to existing multi-focus interaction techniques such as rubbersheet displays [26], accordion drawing [21, 31], and space-folding [10]. The tradeoff with stack zooming in comparison to these is that the competing techniques generally provide integrated focus and context, whereas stack zooming lays out focus points as separate viewports on the visual space. It may even be argued that stack zooming is not a true multi-focus interaction technique because of this. However, our intention with the stack zooming design was to provide a distortion-free visual display that retains the familiarity of the original visual representations drawn on the display, and that is what distinguishes the technique from its competitors. Of course, we would be interested in studying how our technique fares in comparison to these techniques.

7 Conclusions and Future Work

We have presented a theoretical background and a practical implementation of a multi-focus interaction technique called *stack zooming* that integrates multiple focus points in time-series dataset visualizations with their respective context and relationships. We have also shown how the zoom hierarchy built during visual exploration can serve as an input to presenting the insights gained during a particular session. Our prototype is built using Java and showcases the basic functionality of the stack zooming technique for medium-sized temporal datasets. To further press home our contributions, we have exemplified the tool's use for exploring stock market crashes in a financial dataset consisting of several market indices for a period of more than 10 years.

Our future work will entail studying the performance of the tool in comparison to similar tools, such as LiveRAC [19], Continuum [4], and the Mélange [10] space-folding technique. We would also like to improve the tool to better support collaborative visual exploration settings involving teams of analysts working together, and study how the tool can help analysts fill different roles in the analysis process. Finally, we are interested in exploring similar applications of communication-minded visualization [35] in our future endeavors.

References

[1] W. Aigner, S. Miksch, W. Müller, H. Schumann, and C. Tominski. Visualizing time-oriented data—a systematic view. *Computers and Graphics*, 31(3):401–409, 2007.

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

[3] R. A. Amar, J. Eagan, and J. T. Stasko. Low-level components of analytic activity in information visualization. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 111–117, 2005.

[4] P. André, M. Wilson, A. Russell, D. A. Smith, A. Owens, and M. C. Schraefel. Continuum: designing timelines for hierarchies, relationships and scale. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 101–110, 2007.

[5] B. B. Bederson, J. Grosjean, and J. Meyer. Toolkit design for interactive structured graphics. *IEEE Transactions on Software Engineering*, 30(8):535–546, 2004.

[6] J. T. Biehl, M. Czerwinski, G. Smith, and G. G. Robertson. FASTDash: a visual dashboard for fostering awareness in software teams. In *Proceedings of the ACM CHI 2007 Conference on Human Factors in Computing Systems*, pages 1313–1322, 2007.

[7] K. Brodlie, A. Poon, H. Wright, L. Brankin, G. Banecki, and A. Gay. GRASPARC: A problem solving environment integrating computation and visualization. In *Proceedings of the IEEE Conference on Visualization*, pages 102–109, 1993.

[8] A. J. B. Brush, B. Meyers, D. S. Tan, and M. Czerwinski. Understanding memory triggers for task tracking. In *Proceedings of the ACM CHI 2007 Conference on Human Factors in Computing Systems*, pages 947–950, 2007.

[9] S.-M. Chan, L. Xiao, J. Gerth, and P. Hanrahan. Maintaining interactivity while exploring massive time series. In *Proceedings of the IEEE Symposium on Visual Analytics Science & Technology*, pages 59–66, 2008.

[10] N. Elmqvist, N. Henry, Y. Riche, and J.-D. Fekete. Mélange: Space folding for multi-focus interaction. In *Proceedings of the ACM CHI 2008*

Conference on Human Factors in Computing Systems, pages 1333–1342, 2008.

[11] N. Elmqvist and P. Tsigas. Animated visualization of causal relations through growing 2D geometry. *Information Visualization*, 3:154–172, 2004.

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

[13] G. W. Furnas. Generalized fisheye views. In *Proceedings of the ACM CHI'86 Conference on Human Factors in Computer Systems*, pages 16–23, 1986.

[14] G. W. Furnas. A fisheye follow-up: further reflections on focus + context. In *Proceedings of the ACM CHI 2006 Conference on Human Factors in Computing Systems*, pages 999–1008, 2006.

[15] J. Heer, J. D. Mackinlay, C. Stolte, and M. Agrawala. Graphical histories for visualization: Supporting analysis, communication, and evaluation. *IEEE Transactions on Visualization and Computer Graphics (Proc. InfoVis'08)*, 14(6):1189–1196, 2008.

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

[17] Z. Liu, N. J. Nersessian, and J. T. Stasko. Distributed cognition as a theoretical framework for information visualization. *IEEE Transactions of Visualization and Computer Graphics*, 14(6):1173–1180, 2008.

[18] J. D. Mackinlay, G. G. Robertson, and S. K. Card. The Perspective Wall: Detail and context smoothly integrated. In *Proceedings of the ACM CHI'91 Conference on Human Factors in Computing Systems*, pages 173–179, 1991.

[19] P. McLachlan, T. Munzner, E. Koutsofios, and S. C. North. LiveRAC: interactive visual exploration of system management time-series data. In *Proceedings of the ACM CHI 2008 Conference on Human Factors in Computing Systems*, pages 1483–1492, 2008.

[20] W. Müller and H. Schumann. Visualization methods for time-oriented data—an overview. In *Proceedings of the Winter Simulation Conference*, pages 737–745, 2003.

[21] T. Munzner, F. Guimbretière, S. Tasiran, L. Zhang, and Y. Zhou. TreeJuxtaposer: scalable tree comparison using focus+context with guaranteed visibility. In *Computer Graphics (ACM SIGGRAPH 2003 Proceedings)*, pages 453–462, 2003.

[22] C. Plaisant, B. Milash, A. Rose, S. Widoff, and B. Shneiderman. LifeLines: Visualizing personal histories. In *Proceedings of the ACM CHI'96 Conference on Human Factors in Computing Systems*, pages 221–227, 1996.

[23] H. Richter, J. A. Brotherton, G. D. Abowd, and K. N. Truong. A multiscale timeline slider for stream visualization. Technical Report GVU-99-30, GVU Center, Georgia Institute of Technology, July 2006.

[24] L. Robert and E. Lecolinet. Browsing hyperdocuments with multiple focus+context views. In *Proceedings of the ACM Conference on Hypertext*, pages 293–294, 1998.

[25] D. M. Russell, M. J. Stefik, P. Pirolli, and S. K. Card. The cost structure of sensemaking. In *Proceedings of the ACM INTERCHI'93 Conference on Human Factors in Computing Systems*, pages 269–276, 1993.

[26] M. Sarkar, S. S. Snibbe, O. J. Tversky, and S. P. Reiss. Stretching the rubber sheet: A metaphor for visualizing large layouts on small screens. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 81–91, 1993.

[27] B. Shneiderman. Tree visualization with tree-maps: A 2-D space-filling approach. *ACM Transactions on Graphics*, 11(1):92–99, Jan. 1992.

[28] G. Shoemaker and C. Gutwin. Supporting multi-point interaction in visual workspaces. In *Proceedings of the ACM CHI 2007 Conference on Human Factors in Computing Systems*, pages 999–1008, 2007.

[29] Y. Shrinivasan and J. van Wijk. Supporting the analytical reasoning process in information visualization. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 1237–1246, 2008.

[30] S. F. Silva and T. Catarci. Visualization of linear time-oriented data: a survey. In *Proceedings of the First International Conference on Web Information Systems Engineering*, pages 310–, 2000.

[31] J. Slack, K. Hildebrand, and T. Munzner. PRISAD: A partitioned rendering infrastructure for scalable accordion drawing (extended version). *Information Visualization*, 5(2):137–151, 2006.

[32] C. Stolte and P. Hanrahan. Polaris: a system for query, analysis and visualization of multi-dimensional relational databases. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 5–14, 2000.

[33] J. J. Thomas and K. A. Cook, editors. *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. IEEE Press, 2005.

[34] E. R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, Connecticut, 1983.

[35] F. B. Viégas and M. Wattenberg. Communication-minded visualization: A call to action. *IBM Systems Journal*, 45(4):801–812, Apr. 2006.

[36] T. D. Wang, C. Plaisant, A. J. Quinn, R. Stanchak, S. Murphy, and B. Shneiderman. Aligning temporal data by sentinel events: discovering patterns in electronic health records. In *Proceedings of the ACM CHI 2008 Conference on Human Factors in Computing Systems*, pages 457–466, 2008.