Stack Zooming for Multifocus Interaction in Skewed-Aspect Visual Spaces

Waqas Javed, *Student Member*, *IEEE*, and Niklas Elmqvist, *Member*, *IEEE*

Abstract—Many 2D visual spaces have a virtually one-dimensional nature with very high aspect ratio between the dimensions: examples include time-series data, multimedia data such as sound or video, text documents, and bipartite graphs. Common among these is that the space can become very large, e.g., temperature measurements could span a long time period, surveillance video could cover entire days or weeks, and documents can have thousands of pages. Many analysis tasks for such spaces require several foci while retaining context and distance awareness. In this extended version of our IEEE PacificVis 2010 paper, we introduce a method for supporting this kind of multifocus interaction that we call *stack zooming*. The approach is based on 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 exploration. Correlation graphics show the relation between stacks and strips of different levels, providing context and distance awareness for the foci. The zoom hierarchies can also be used as graphical histories and for communicating insights to stakeholders and can be further extended with annotation and integrated statistics.

Index Terms—Multifocus interaction, temporal data, comparative visualization, visual exploration, visual analytics, interaction

1 Introduction

Large visual spaces are becoming increasingly common as we apply modern information technology to the data deluge facing our society today; examples include online maps, handheld GPS navigation systems, graphical documents, and so on. A considerable portion of these visual spaces can be regarded as *skewed-aspect* spaces: While they may have a small and constant extension in a second dimension, they extend primarily in a single dimension. Examples of such visual spaces include visualizations of time-series data (such as temperature measurements or stock values over time), graphical representations of multimedia data (such as sound or video streams), or long documents of text and images.

Navigating in this kind of visual spaces is challenging due to the unbalanced ratio between the dimensions [2], the difficulty of creating an overview [3], and the arbitrarily large scale of the primary dimension of the space. In particular, this large scale—days or weeks of surveillance video, years of time-series data, or hundreds of pages in a document—means that a user performing some analytical task in the space may have to spend considerable time panning and zooming between different areas of interest, a tedious, error prone, and potentially disorienting process [4]. In this paper, we argue that such tasks requires *multifocus interaction* [5]—the capability to simultaneously view several parts of a space at high detail while retaining awareness of context and spatial relations.

Consider, for example, a stock market analyst who is trying to predict future market trends using line graphs of stock market indices by extrapolating from several similar situations at different times in the past. Multifocus interaction allows the analyst to see 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. However, most existing temporal visualizations—for example, TimeSearcher [6], ATLAS [7], and LifeLines [8]—do not support the requirements of multifocus interaction.

In this extended version of our IEEE PacificVis 2010 paper [1], we present *stack zooming*, an overview+detail technique for multifocus interaction in skewed-aspect visual spaces—which includes the temporal data set examples discussed above. The technique is based on stacking strips of 1D data into a *stack hierarchy* and showing their relations. We discuss the general approach for implementing stack zooming for any skewed-aspect space, regardless of the visual representation and application used. To exemplify the new method, we also present several concrete applications of stack zooming. We then validate the work with a case study deploying the technique for network management, as well as with a quantitative experiment comparing its efficiency to classic overview+detail methods.

The contributions of this paper are the following:

1. the general stack zooming technique for skewed-aspect visual spaces;
2. four applications of stack zooming for different domains;
3. a case study for log analysis involving analysts in a network management setting; and
4. results from a controlled experiment evaluating the utility of stack zooming compared to standard overview+detail layout.

The latter three (2-4) of these contributions are new to the journal version of the work, and thus significantly extends the work in the IEEE PacificVis 2010 paper [1].

• *The authors are with the School of Electrical and Computer Engineering, Purdue University, 465 Northwestern Avenue, West Lafayette, IN 47907. E-mail: {wjaved, elm}@purdue.edu.*

Manuscript received 4 Apr. 2012; revised 25 Sept. 2012; accepted 11 Dec. 2012; published online 21 Dec. 2012.
Recommended for acceptance by M. Chen.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2012-04-0061.
Digital Object Identifier no. 10.1109/TVCG.2012.323.

2 Motivation

We define a *skewed-aspect visual space* as a space consisting of a primary and a secondary dimension. The term is derived from the potentially very large aspect ratio caused by the primary dimension becoming very long, whereas the secondary is typically much shorter. Examples of primary dimensions include time for temporal data, page sequence for documents, and temporal sequence for multimedia streams. The secondary dimension, on the other hand, either has a fixed size or is space-filling, meaning that it can be adapted to whatever size it is allocated.

Because of this, the aspect ratio of this kind of visual space can also either be fixed or space filling. This gives rise to some complications when allocating display space to the visual space. For example, for a time-series visualization using line graphs, the primary dimension is time, and convention usually assigns this to the horizontal (X) dimension. The secondary dimension—the vertical size allocation for the graph—can be adapted to whatever space is available (i.e., it is space filling). For a document, the primary dimension is the length of the document (by convention usually assigned to the vertical dimension) and the secondary dimension (the width of the document) is fixed. In other words, for documents, the aspect ratio of the space must be fixed to avoid distortion.

Many analytical tasks for large visual spaces involve correlation between several areas of interest. We call an area of interest a *focus* or *focus region*. Users generally need to be able to view all focus regions at high resolution, while retaining an awareness of the context surrounding each region, the context between the regions, and the relative spatial relations of the focus regions. This style of interaction is generally known as *multifocus interaction* [5].

2.1 Problem Statement

While multiscale navigation is challenging for all large visual spaces [9], the lopsided relation between the primary dimension and secondary dimension for skewed-aspect visual spaces makes navigation particularly difficult. Consider the length (primary) and width (secondary) dimensions of a text document; the latter is fixed, whereas the former is not. Therefore, unlike for a 2D visual space like a map, it is typically not possible to attain a useful overview [3] of a skewed-aspect visual space—at least not without relaxing the linear layout of the primary dimension, such as in the space-filling thumbnail approach for document browsing proposed by Cockburn et al. [10]. Even for visual spaces where the secondary dimension is space filling, such as for line graphs of time-series data, the skewed aspect ratio between the dimensions may cause misperceptions [2].

Because primary and secondary dimensions are decoupled in skewed-aspect visual spaces, we argue that multifocus interaction tasks are particularly difficult to perform in this setting. The user typically has to resort to one of the following strategies:

- *Navigation:* Pan and zoom between the areas of interest, a tedious and error-prone process [4]; or
- *Split-screen:* Splitting the screen into multiple views, one for each focus region. Context and spatial relations between focus regions are lost.

2.2 Design Goals

While existing multifocus techniques such as space folding [5] and multiple fisheye views [11] are generally focus+context techniques that use distortion, this is actually not a hard requirement. Actually, distortion can sometimes be confusing because it introduces nonlinear elements into the display, elements that are typically not visually stable under translation [12]. Therefore, we include an additional design goal (G5) beyond those of standard multifocus interaction [5]:

TABLE 1
Multifocus Interaction Design Goals Supported by Existing Interaction and Navigation Techniques

Technique	G1	G2	G3	G4	G5	Examples
Pan + zoom	–	–	–	–	✓	[13], [14]
Fisheye views	✓	✓	✓	–	–	[4]
Rubber sheet	✓	✓	✓	✓	–	[15]
Split-screen	✓	✓	–	–	–	[11]
Overview+detail	–	✓	✓	✓	✓	[16], [17]
Space folding	✓	✓	✓	✓	–	[5]
Multi-resolution	–	✓	✓	✓	✓	[18]
Hierarchical zoom	✓	✓	✓	–	✓	[19]

G1. *Multiple foci:* guaranteed visibility of all focus regions at independent levels of zoom;
G2. *Surrounding context:* show as much as possible of the space surrounding each focus region;
G3. *Intervening context:* convey an awareness of the space between focus regions;
G4. *Spatial relation:* communicate the relative distance and position of the foci on the space; and
G5. *No distortion:* the display should not be nonlinearly distorted and should be visually stable.

In the rest of this paper, we will discuss a novel approach to achieving these design goals for the skewed-aspect visual spaces discussed here.

3 Related Work

Multifocus interaction [5] is a conceptual framework that integrates multiple focus+context [4] views with guaranteed visibility to support both focus, context, and spatial awareness. There are many ways to support multifocus interaction (see Table 1):

- *Standard navigation techniques* like pan and zoom, generally requiring repeated interaction;
- *Focus+context techniques* that integrate the foci in the context of the space as a whole;
- *Split-screen techniques* where the viewport is partitioned into several smaller viewports that each show different parts of the space;
- *Overview+detail techniques* where a bird's-eye view shows the overall context and a detail view shows the focus region; and
- *Hierarchical navigation techniques* that allow for progressive panning and zooming.

These are all described in the following sections.

3.1 Standard Navigation Techniques

Panning and zooming are the most basic of navigation operations for large visual spaces [9] and can be found in virtually all graphical applications such as map viewers, document editors, and web browsers. However, basic pan

and zoom do not directly support any of our above design goals (except G5) [5].

More advanced panning and zooming interactions exist that achieve better performance, such as combining panning and zooming [9], [20], coupling zoom factor to scrolling speed [14], and using a second dimension for zoom speed [13].

3.2 Focus+Context Techniques

Focus+context [4] is one particularly powerful approach for seeing details while maintaining overview where views of the focus regions are integrated into their surrounding context. This lets the user to see foci in direct relation to their context, but requires space to be distorted (violating G5).

The most widely known focus+context techniques are fisheye views [4], including techniques for multiple fisheyes [11], and the rubber-sheet metaphor [15]. The latter class of techniques are particularly relevant because they allow for freely deforming space to guarantee visibility of several focus points, and is the approach taken by the LiveRAC system [21].

Space folding [5] is a recent alternative approach that is similar to rubber-sheet techniques, but which explicitly folds space away into 3D instead of compressing or stretching it. The visual representation of the folded space better supports the spatial relation (G4) design goal than rubber sheet methods.

However, all of the above techniques are based on nonlinear space distortion, which can be nonintuitive, visually unstable, and difficult to understand [12]. To combat these issues, Zanella et al. [22] studied how to add visual cues to decrease the impact of distortion, and Gutwin [12] dynamically adapted the distortion depending on the user's interaction. Most recently, the Sigma lens framework [23] generalizes focus+context lenses to other dimensions beyond space, such as time and translucence; this allows for supporting the no distortion (G5) design goal as well.

3.3 Split-Screen Techniques

A straightforward way to support multiple focus regions (G1) is simply to create a separate viewport for each of the foci, i.e., to split the screen, a common approach in desktop applications. Shoemaker and Gutwin [11] present a multifocus technique that automatically splits the screen when the focus points are moved too far apart. However, standard split-screen techniques provide no awareness of the intervening context or spatial relation between the focus regions.

3.4 Overview+Detail Techniques

A specialization of split-screen techniques is overview+detail [16] techniques that provide both a view of the current focus region as well as a bird's-eye view of the surrounding context of the focus (often with a visual indication showing where the focus is located in the context). This requires the user to split their attention between several viewports, but allows for showing detail without distortion.

Hornbæk and Frøkjær [17] showed that overview+detail can outperform focus+context [4] techniques in some situations, but most existing overview+detail implementations do not support multiple focus points (G1). For example, the continuum faceted timeline browser [24] supports many multifocus interaction goals, but provides only one level of overview of the timeline, meaning that detail and context awareness are limited.

3.5 Hierarchical Navigation Techniques

Two existing systems are of particular relevance to the stack zooming technique presented in this work. The multiresolution time slider for multimedia data presented by Richter et al. [18] uses a hierarchical zoom stack similar to ours. However, theirs is primarily an interaction technique for selecting single time periods and does not support multiple focus points.

Second, the multipage zooming technique presented by Robert and Lecolinet [19] defines a hierarchical zooming technique similar to our stack zooming technique. However, their technique is applied to web browsers and uses a node-link diagram to show the zoom hierarchy that does not communicate the spatial relation of focus regions (G4), compared to the explicit spatial hierarchy representation that we use.

4 Designing Stack Zooming

Stack zooming is a hierarchical overview+detail technique that supports multifocus interaction as follows:

- G1. *Multiple segments:* display space is split among multiple segments, or *strips*, of the visual space;
- G2. *Subset context:* segments show the context around each focus point;
- G3. *Overall context:* focus regions are arranged in a hierarchy so that the full space is always visible;
- G4. *Correlation:* visual cues correlate one segment to its children, showing their spatial relations; and
- G5. *No distortion:* the hierarchical overview+detail approach means that distortion is not necessary.

When the user begins to analyze a temporal visualization using stack zooming, the whole display is taken up by the full time series on a single (main) strip. Dragging on the surface of this strip, the user can create a child strip of the main strip that displays the selected subset of the data. The size of the created strip thus directly controls the zoom level. Further zoom operations on the main 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 is a focus region and is guaranteed to be visible.

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 pans the child strip, and children can be panned directly by using the arrow keys. In this way, users can quickly explore the temporal data set with a sequence of simple zoom and pans while retaining multifocus support.

4.1 Visual Space

Stack zooming is just a space management technique and can be applied to other visual spaces such as multimedia streams, bipartite graphs, or text documents.

Furthermore, although in this section, we will use a horizontal layout where stacks are layered on top of each other, stack zooming can also just as easily use a vertical layout arrangement. In this setup, stacks are arranged in vertical columns, and strips split each column into horizontal rows. The appropriate layout orientation depends on the application; for a timeline, a horizontal layout works best, whereas for a document viewer, vertical layout is more appropriate.

4.2 Zoom Stacks

The basic structure in stack zooming is a hierarchical *zoom stack* (note that zoom stacks are trees and not lists like normal stacks). Just like any other tree, a zoom stack is defined by a single root node r containing all of the nodes of the tree. The zoom stack supports basic tree operations such as finding the depth of individual nodes, the depth of the whole tree, the number of children, as well as all standard traversals.

Nodes in a zoom stack are called *zoom nodes*. A single zoom node captures a single *strip* (a focus region in the 1D visual space) in the stack zooming technique. Thus, the node consists of a range $[e_0, e_1]$ describing the extents of the focus region in the primary dimension of the visual space, a layout allocation for this particular node on the screen (width or height, depending on the orientation of the space), a parent node, and an ordered list of child nodes.

Screen allocations are specified as normalized 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 drawn.

4.3 Layout

Nodes in a zoom stack are laid out on the visual 2D substrate using a space-filling layout (Fig. 2) that splits the vertical space by the depth of the tree (assigning an equal amount to each tree level), and the horizontal space by the number of siblings for each level of the tree (assigning an equal amount to each sibling). As discussed above, the layout may also be rotated, using horizontal space for stacking layers in the tree and vertical space for siblings for each level.

There is naturally a perceptual limit to how many layers and strips can be added to the screen before individual segments are no longer legible. This limit is proportional to the amount of display space allocated to the whole stack zooming canvas. One approach to manage this problem is to introduce scrolling for both spatial dimensions, but this means that visibility of layers and segments will be violated. Another approach is to dynamically resize the dimensions of segments depending on interaction history: for example, segments or layers that were not clicked recently could be smoothly shrunk to minimal size to give more space to segments that are currently in focus.

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 panning focus points, is a special case—see Section 5.4.

To make the tree structure of the zoom stack explicit, one design alternative is to not divide space equally across siblings of each level of the zoom stack, but rather to assign space to whole subtrees (somewhat similar to the node-link approach taken by Robert and Lecolinet [19]). This would mean that each child would have to stay within the extents of its parent. It would also give a visual indication of the parent-child relationships between strips in adjacent levels, and thus decrease the need for explicit correlation graphics (discussed next). However, because visual exploration using stack zooming often results in unbalanced zoom stacks, this design would result in suboptimal use of the available screen space. Therefore, global space allocation across each level is generally the better design alternative.

4.4 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 relationship between parent strips and child strips in adjacent levels of the zoom stack. However, as argued above, we cannot directly show ancestor relationships in the layout using proximity cues, or we will waste valuable screen space. Therefore, we introduce *correlation graphics* that visually indicate these relationships (Fig. 2):

- *Color-coded zoom areas:* Parent strips show color-coded (but semitransparent) selection areas that indicate the position and extents of each child strip in the time series.
- *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:* Arrows show relations from zoom areas in parents to the respective children.
- *Brushing:* Hovering the pointer over a focus region will highlight (e.g., using color or shadows) the entire ancestor hierarchy above the focus.

5 Implementing Stack Zooming

Many graphical applications, such as digital maps, visualization applications, and photo viewers, support successive zooming operations. The difference for stack zooming is that successive zoom operations do not exchange the old view with the new view, but, instead, *all* views are kept on screen at all times. The layout algorithm, which manages the space allocation of each view, is therefore central to stack zooming.

5.1 Layout

The layout algorithm uses 2D geometric space to create a visual hierarchy of zoomed regions. Depending on whether the arrangement is vertical or horizontal, the algorithm will use the *dominant axis*—X for vertical layouts, Y for horizontal ones—for layers, and the other, *nondominant*, axis for siblings in each layer. Given an skewed-aspect space and a zoom stack (a hierarchy of focus points, i.e.,

position and extents) in this space, the layout algorithm proceeds as follows:

1. *Layer allocation:* Allocate display space in the dominant axis to layers by splitting the available space by the number of zoom stack levels.
2. *Main allocation:* Allocate the full extents of the nondominant axis to the top level zoom region that shows the primary axis of the visual space.
3. *Strip allocation:* For each layer, allocate display space along the nondominant axis to siblings by splitting the available space by the number of siblings on this level in the zoom stack.

These space allocations may optionally be controlled by the user (typically by dragging the mouse on layer and sibling borders) so that they are only default values when a new layer or sibling is created.

5.2 Reshaping Strips

One practical problem that arises as an effect of the above layout algorithm is that the aspect ratio of individual zoom regions will change dynamically as layers and strips are added to the zoom stack. For visual spaces with a space-filling secondary dimension, like a time-series visualization, this is typically no problem; the visualization will reshape to whatever geometry it is given (although aspect ratios are often important for correctly interpreting trends in line graph visualizations [2]). However, when the secondary dimension is fixed, such as for a document or a video stream, changing the aspect ratio will cause deformations (interfering with design guideline G5).

Fig. 3 shows three different reshaping strategies, characterized by whether the aspect ratio and the extents of the focus regions are kept fixed. The leftmost alternative, with fixed extents and variable aspect ratio, is the default and works well for most visualizations. For when the aspect ratio must be kept fixed to avoid deformation (G5), either the extents can be changed to suit the aspect ratio (center alternative in Fig. 3) or the whole region can be uniformly scaled to fit available display space (right in Fig. 3).

Adopting the changing extent approach means that it is not always possible to show everything selected by the user using a bounding box selection. In situations like this, it is better to adopt a line or point-based selection and then fit as much data as possible, while keeping the aspect ratio constant. Bounding box selection can be used with the third approach, i.e., using uniform scaling to achieve both fixed aspect ratio and fixed extent. However, this may results in not making full use of the available space. Which strategy works best depends on the application or could be left in the hands of the user in a practical implementation.

5.3 Navigation

The zoom operation is intrinsic to stack zooming and creates new focus regions. It is therefore not a navigation operation in itself. Instead, the main navigation operation for stack zooming is *panning* a focus point. Panning can either be done by dragging the zoom area selections in a parent strip or by panning a child strip directly (arrow keys or mouse).

5.4 Overlapping and Merging

Panning a child strip may give rise to a special layout case when the interval covered by one strip overlaps that of another strip. During overlap, the layout will not be changed to maintain stability of the display (G5), 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. Overlapping reinforces the awareness of a user navigating in the time series by merging adjacent child strips when their intervals overlap. Fig. 4 shows an overview of this operation.

However, strip merging requires that all zoom strips in a stack level cover the same length of interval (i.e., all strips should use the same zoom factor), or visual distortion will result (violating G5). Therefore, it may not be practical for all applications. An alternative solution is to let strips overlap and merely replicate the visual representation on all overlapped strips. In such cases, strips will be ordered according to their minimum boundary (e_0).

6 Applications

We present four novel applications of stack zooming in different domains and contexts:

- *TraXplorer:* a time-series visualization tool;
- *SZ-Timeline:* a time-series visualization component for the Web that uses JavaScript and scalable vector graphics (SVG);
- *Hugin-TraXplorer:* a collaborative visualization designed for digital tabletops; and
- *PDF-StackZoom:* a stack-zooming PDF viewer.

We will describe each of these applications below.

6.1 Time-Series Visualization: TraXplorer

The TRAXPLORER system is a time-series visualization tool supporting multifocus interaction using the stack zooming technique introduced in this paper (Fig. 1). Time series are represented as multiple *tracks*, hence the name of the tool.

6.1.1 Related Work: Time-Series Visualization

Much research has been published related to visualization of time-series data sets; see Aigner et al.'s survey [25]. During this time, temporal visualization has evolved from basic timeline graphs (some of these dating back hundreds of years) to sophisticated visualization systems designed for various purposes.

The Perspective Wall [26] presents temporal data using a 3D rendering that incorporates a natural focus+context [4] perspective distortion. LifeLines [8] was one of the early systems that used visualization to explore discrete events in personal histories. TimeSearcher [6] is a time-series visualization tool that uses timeboxes to generate visual queries to explore the data sets. Continuum [24] is a Web 2.0 tool for faceted timeline browsing. ATLAS [7] is a system for visualizing massive time-series data sets. Most recently, LiveRAC [21] is a multifocus visual exploration tool for time-series data in system management.

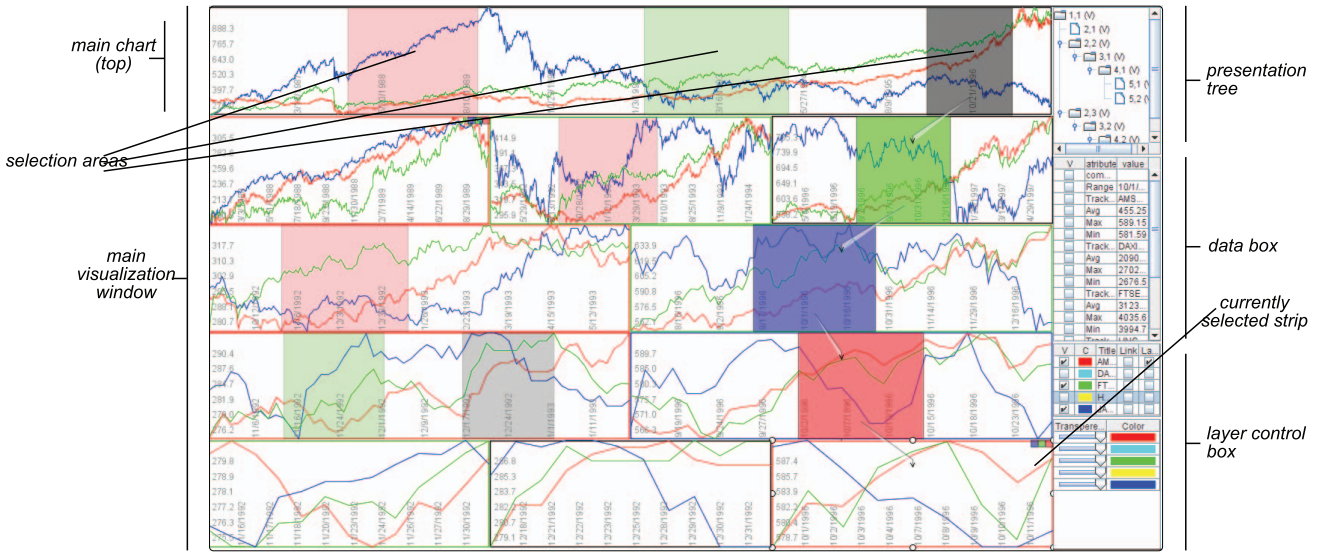


Fig. 1. Stack zooming in a stock market data set using TraXplorer (one of our applications of stack zooming). The line graph segments are arranged in a zoom stack that was built during visual exploration.

6.1.2 Visual Exploration

The TraXplorer exploration interface (Fig. 1) consists of the following components:

- *Visualization window:* The main visualization window is a visual space supporting stack zooming. It supports visualizations of time-series data on a common time axis and on potentially different value axes. The type of visual representation in the visualization window is independent of the layout management—our implementation currently supports basic line graphs, filled area charts, and horizon graphs [27] (the first one overlaid, the latter two juxtaposed [28]).
- *Data box:* This interface component gives local statistics about the currently selected strip in the zoom stack. This provides details-on-demand for computing measures such as minimum, maximum, average, median, and standard deviation metrics for a particular track.
- *Layer control:* The layer control box moves, deletes, and toggles the visibility of individual tracks, as well as to change color mapping, transparency, and track title. Furthermore, using the layer control, tracks can be *linked* to use the same scale for the value (Y) axis, thereby supporting direct comparison across tracks.

The visualization window supports the main stack zooming interactions. Dragging the mouse on a visualization strip will create a child strip, which can be panned by moving the selection or by using arrow keys inside the focus region. Each strip can also be maximized, hidden, and deleted. Deleting a strip will delete all of its children. Furthermore, dragging the border of a strip enables resizing its space allocation.

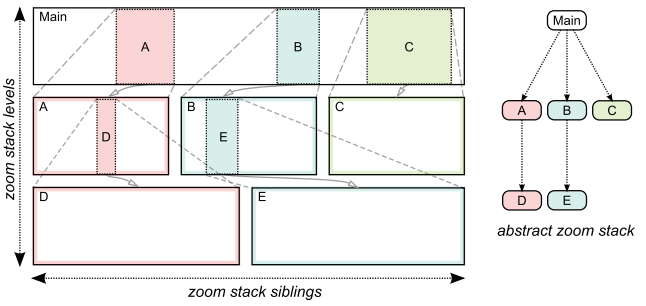


Fig. 2. General layout mechanism for stack zooming. Color coding show parent-child relationships, and correlation graphics make the relations explicit.

6.1.3 Implementation Notes

The TRAXPLORER system was implemented in Java using the Piccolo structured 2D graphics toolkit [29], [30]. The key components in our implementation include the time strip class (implemented as subclasses of Piccolo's `PNode` basic scene graph node class), the visual representations, and the layout manager.

6.2 Stack Zooming on the Web: SZ-Timeline

SZ-TIMELINE is a stack zooming implementation for the web using JavaScript and SVG for dynamic vector graphics in the browser (Fig. 5). Implemented as a Google Visualization API[1] component, this application is publically available on the web[2] and can be applied to any standard online data source to display time-series data in visualization mashups.

6.2.1 Related Work: Visualization on the Web

Visualization on the web allows for social data analysis in asynchronous, distributed collaboration settings [31]. It has become particularly important with the advent of information visualization for the masses, primarily through projects like NameVoyager [32], Sense.us [33], and Many Eyes [34].

There are many different alternatives for building visualizations for the web, ranging from classic Java applets, to more recent RIA technologies like Adobe

1. http://code.google.com/apis/visualization/.
2. https://engineering.purdue.edu/~elm/projects/gvis/.

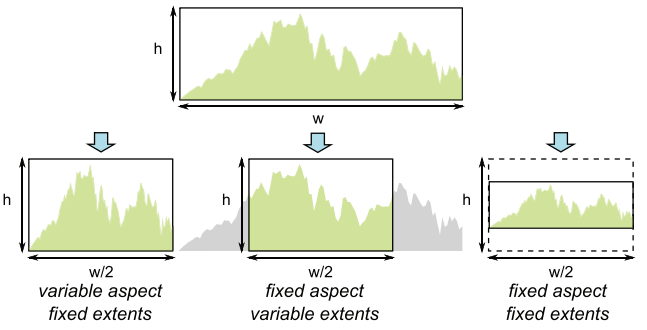


Fig. 3. Three different strategies for reshaping strips. The appropriate strategy depends on the visual space.

Flash/Flex and JavaScript coupled with SVG. A number of visualization toolkits have also been developed for these platforms, including the Google Visualization API, the Java InfoVis Toolkit,[3] and D3 [35].

6.2.2 Visual Representations

Utilizing the fact that stack zooming is a space management technique that is independent of the structure of the visual space, our JavaScript/SVG stack zooming timeline supports several different visual representations. Beyond standard line graphs, we have also implemented juxtaposed filled area charts, horizon graphs [27], and braided graphs [28].

6.2.3 Implementation Notes

As discussed above, our implementation is built in JavaScript and uses SVG for rendering vector graphics directly into the web browser. Because some web browsers do not directly support SVG rendering (notably the current version of Microsoft Explorer, version 8), we use the Raphaël[4] JavaScript library, which transparently uses vector markup language (VML) for these browsers instead. With full stack zooming support and four different visual representations, our implementation is merely 182 lines of commented code.

6.3 Collaborative Visualization: Hugin-TraXplorer

Beyond the asynchronous and distributed collaboration supported by the SZ-Timeline component, we have also built a stack zooming implementation called HUGIN-TRAXPLORER for synchronous collaboration on direct-touch tabletop displays [36]. Our focus here is on the collaborative aspects, and thus, this implementation only supports line graph representations.

6.3.1 Related Work: Collaborative Visualization

Collaborative visualization is gradually becoming a necessity to better support real-world problem-solving and decision-making processes, and results point to the benefits of collaborative data analysis (e.g., [37]). Furthermore, new computer platforms—like wall-sized and tabletop displays—open up exciting new possibilities for collaborative visualization. Existing work already apply visualization to synchronous and colocated collaboration on large direct-touch displays [38], [39], but the potential is huge and much work remains to be done here.

3. http://thejit.org/.
4. http://raphaeljs.com/.

6.3.2 Collaborative Stack Zooming

Hugin-TraXplorer is designed for synchronous collaborative analysis on multitouch tabletops. Studies have shown that effective colocated collaborative visualization requires multiple coordinated views to allow for changing collaboration styles [39]. Therefore, we design our implementation to have a single shared timeline, and then allow participants to create a linked personal interaction workspace where they can view and interact with the data (Fig. 6).

Individual workspaces can be scaled, resized, and moved using standard multitouch gestures. Because our current hardware platform—a two-projector multitouch tabletop display built in our laboratory—does not explicitly support user identity tracking, we rely on social protocols for protecting workspaces [40]. The following stack zooming operations are supported on the time-series data inside a workspace:

- *Create focus region:* Users can create a focus region using a two-point gesture inside a strip.
- *Pan focus region:* Dragging on a selection area (Fig. 6) will pan the associated focus region.
- *Delete focus region:* Double tapping on a selection area will remove the focus region.

Our implementation also supports awareness and coordination mechanisms [39] such as overviews, telefingers (remote touches appearing as "ghost" fingers on the local tabletop surface), and access control.

6.3.3 Implementation Notes

Just like the original TraXplorer application, Hugin-TraXplorer is built in Java and uses the Piccolo [29] structured 2D graphics library. The collaborative version was built from scratch to fully support multitouch interaction and multiple concurrent users.

6.4 Document Navigation: PDF-StackZoom

Reading a digital document on a computer screen is challenging due to many reasons [41], an important one being the difficulty of navigating within the document. Many tasks require correlating several different parts of a document, i.e., multifocus interaction tasks.

To support these operations, we have designed PDF-StackZoom (Fig. 7), a prototype PDF document viewer that supports multifocus interaction within different portions of a document.

6.4.1 Related Work: Document Navigation

Over the years, many researchers have proposed different methods to improve navigation through digital documents [10], [42]. However, we are aware of no research or commercial document viewers that support multiple focus areas within a document.

6.4.2 Multifocus Document Navigation

Unlike the other application examples discussed so far in this paper, the PDF-StackZoom tool uses a vertical layout where layers in the zoom stack are columns instead of rows (see Fig. 7). Furthermore, similar to the continuum browser [24], we only support two levels in the zoom stack: an overview, and a focus.

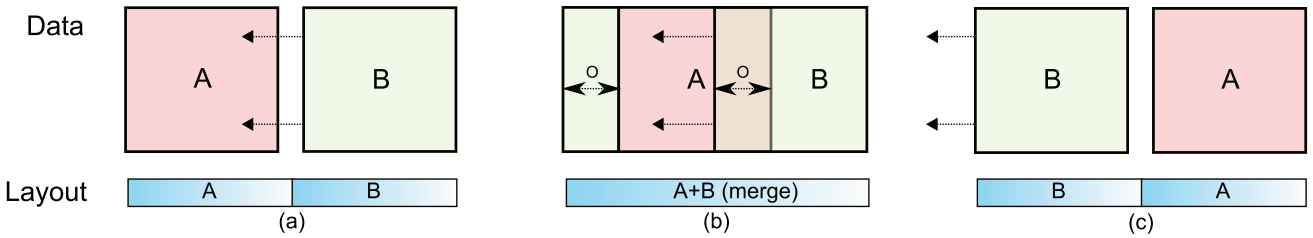


Fig. 4. Strip merging during overlap. (a) Two strips, A and B, of the same width are approaching. (b) Overlap between A and B, causing the layouts to merge into a single strip. (c) There is no longer overlap, so the strips are separate and with correct order in layout space.

Consequently, the application consists of two frames: the overview frame and the focus frame. The overview frame is equivalent to the root node in the zoom stack, and displays the fixed-size thumbnails of all the pages in the current document. The focus frame displays a different focused regions of the document, stacked on one another.

Interaction is a key aspect of multifocus document navigation. New focus regions can be created by simply selecting a rectangular area to view on the overview frame. Panning a focus region is done by either dragging the selection area in the overview frame, grabbing the canvas and dragging in the focus itself or scrolling up or down using the arrow keys when the region has keyboard focus. Similarly, regions can be resized by dragging the border of the selection or region. We also provide ways to both delete and minimize focus regions to permanently or temporarily give more space to other focus regions.

6.4.3 Implementation Notes

PDF-StackZoom was implemented in Java, and uses Piccolo2D [29]. All PDF functionality is provided through the PDF Renderer[5] Java library.

7 Case Study: Network Management

To validate the usefulness of the stack zooming technique, we enlisted the help of the network management group at our local institution. The group administers close to 200 servers and more than 30 Terabytes of storage space. It also manages the webserver cluster, with in excess of 6 million hits per month. Working with the software team in this group, we were able to conduct an in-field case study involving log file analysis using stack zooming.

7.1 Interviews

In the formative stages of this evaluation, we interviewed the members of the group, first in an individual hour-long background session with its manager, and then with two of the senior system analysts at two different occasions. Unlike normal support staff in the network management group, these analysts focus on long-term special projects involving system software, security, and debugging. They are thus very representative of the user group targeted in our work.

Below, we summarize some of the findings we collected from these formative interview sessions:

- *Data:* The most common form of temporal data used by the analysts is log files from the many computer systems managed by the group. These data take one of two forms: either standard system logs, which are time-stamped discrete events (represented by text messages), or quantitative performance data, which show value over time (such as CPU load, bandwidth, disk usage, etc).
- *Application:* The most common use for log data is for diagnostic, debugging, and troubleshooting purposes. The analysts noted that the initiating factor for viewing log data is often some kind of incident of either a security related (intrusion, attack, or scan) or technical (hardware failure, network connectivity, or software fault) nature.
- *Usage:* Log file data are typically studied in an offline and postmortem manner after an incident has happened. The analysts stated that they seldomly used the dynamic log feed, although one of them always had a dedicated window open for this data. The stated purpose of this was to "keep an ear to the ground," i.e., maintain situational awareness of activity on the local network. However, none of the analysts regularly used graphical views of the log data, instead preferring to view (and search in) the raw textual output from system logs. Instead, they

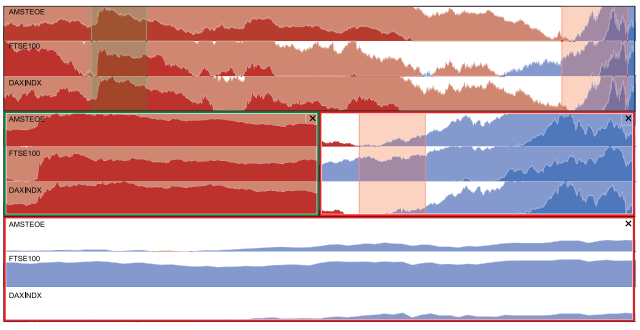


Fig. 5. Stack zooming in JavaScript and SVG for web-based horizon graphs [27] of stock market indices.

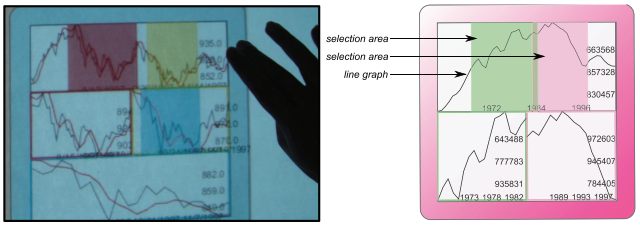


Fig. 6. Time-series visualization using stack zooming designed for collaborative digital tabletops.

5. https://pdf-renderer.dev.java.net/.

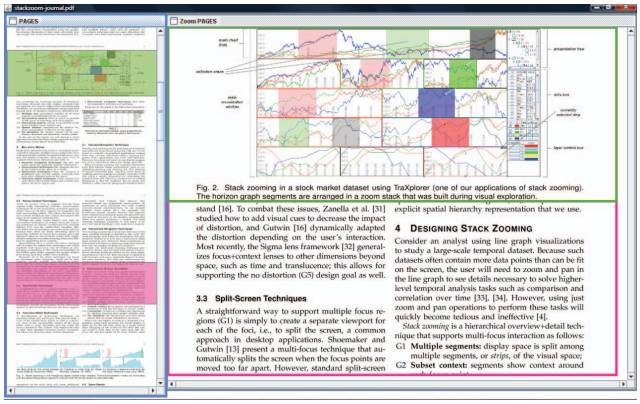


Fig. 7. Stack zooming in a PDF document using the PDF-StackZoom tool. The user has created two focus regions in the document and can move them freely through the visual space defined by the document.

use regular expression filters to color important (or fatal) messages in red or blue so that they stand out.

- *Collaboration:* Our two analysts said that collaboration between them is common, but only for difficult tasks that require expertise from them both. They estimated that 90 percent of tasks could be solved individually. Furthermore, even when collaborating, most collaboration was asynchronous through e-mail even though their offices were within a 1-minute walk of each other. Only in very rare cases (less than 1percent) would they meet up in the same office to solve a problem together.

Thus, we decided to proceed by deploying the SZ-Timeline tool because its web setting best fits the asynchronous work pattern of the two analysts.

7.2 Limited In-Field Deployment

Working with the systems group, we were given access to an excerpt of approximately 24 hours of the system performance logs for all 10 machines (four Apache webservers and six Zope application servers) of the web cluster maintained by the network management group. The log contained average CPU load and network bandwidth consumption (bytes sent per second) synchronously sampled at 1-minute intervals for each machine. All in all, the log data were a comma-separated file containing approximately 1,500 measurements for eight different dimensions.

We loaded these data into a Google Spreadsheet and created an instance of the SZ-Timeline tool to visualize it (the systems group requested we not expose the data or any visualizations due to it potentially being used by would-be attackers). We then sent a link to the tool to the analysts before arranging to interview them about the tool's utility, strengths, and weaknesses.

The overall impression that we received from these interviews was very positive. Analysts commented that they had found it easy to understand how to create new focus points, and remarked on the usefulness of having access to both an overview as well as multiple focus points simultaneously. One analyst noted that this technique made them see visualization "in a new light." The analysts did note that creating multiple focus points quickly diminished the available space allocated to each individual focus point, and requested a way to resize or temporarily minimize particular focus points to make more space for others.

Another concern that the analysts had was that there was no way to create persistent URLs of particular states that they could paste into an e-mail and send to someone else for further analysis. Furthermore, our bare-bones SZ-Timeline implementation does not support adding comments or annotating the visual representation. The analysts thought that this was a necessity for effective collaboration. A production version of SZ-Timeline for network management collaboration would have to include these features, perhaps similar to Many Eyes [34] and Sense.us [33].

Another comment came in regards to the visual representation and the separation of time-series data. SZ-Timeline allows the user to switch between standard line graphs, horizon graphs [27], and braided graphs [28]. The analysts wanted several new ways of aggregating and combining data, such as separate time-series for different machines into different charts, or to overlay different metrics for the same machine in the same chart. This is currently not supported in our implementation.

Unfortunately, our visualization did not give rise to any new insights on behalf of the analysts for the 24-hour logs they studied. Nevertheless, at the end of the interviews, the analysts expressed the desire to get the stack zooming technique implemented into production use. They said that they would very likely use both the visualization and the interaction technique in their daily routine if it was integrated into their standard network management software.

In summary, we collected the following findings for using stack zooming to visualize network log data:

- Maximizing, minimizing, and otherwise manually adjusting layout is an important aspect of stack zooming (supported by TraXplorer);
- Annotating, highlighting, and saving stack zooming layouts would improve collaboration; and
- More complex data management, aggregation, and pivoting is necessary for in-depth analysis.

8 Controlled Experiments

Stack zooming suffers from space limitations as the number of focus region grows, whereas standard overview+detail techniques do not allow multifocus interaction. To shed some light on the advantages and disadvantages of each technique, we performed two controlled user studies designed to compare their performance. Both studies used similar experimental conditions but focused on two different tasks that were chosen as representative of exploring time-series data sets [28]: visual search versus visual comparison.

8.1 Participants

We recruited 12 paid students (nine male and three female) from our university (average age 23) to participate in both user studies. No participant was colorblind and all had normal or corrected-to-normal eye sight. Participation was voluntary and self-selected. All participants had at least

basic computer knowledge and were familiar with simple line graphs.

8.2 Apparatus

Both studies were conducted on a standard desktop computer equipped with a 3-GHz dual-core processor, and running Microsoft Windows XP. The computer had a $19''$ LCD monitor set to $1{,}280 \times 1{,}024$ resolution and a standard two-button mouse was provided to interact with the experimental application. The experimental application was 800×800 pixels in size and positioned at the center of the monitor screen.

8.3 Interaction Technique (T)

The main experimental factor was the technique T:

- *Standard overview+detail (O):* The viewport consists of a detailed view as well as an overview.
- *Stack zooming (S):* The viewport supports stack zooming, allowing the user to create any number of detail views on different levels.

8.4 Software

We implemented a Java application for using both overview+detail and stack zooming to explore a line graph visualization (Fig. 9). Our application supported the basic stack zooming interactions.

For the standard overview+detail technique, we divided the available vertical space equally between the overview and the detail view. This was to avoid unnecessary performance drawbacks because of small overview size, and made this condition essentially the same as stack zooming with only one focus region. For stack zooming, any number of focus regions on any number of levels could be created. For both techniques, a focus region of the size of a single pattern could be created with a single click and moved by dragging its associated highlighted rectangular area across the line graph. Further, focus regions could be deleted by double clicking inside their extents.

8.5 Data Set

A new data set was randomly generated for each trial to avoid learning effects. For each trial, we generated a palette of 32 distinct patterns (small segments of data) and assembled the data set as a random sequence of these patterns. To make the data continuous, we joined two adjacent patterns by using the last data point of a pattern as the starting point for the first point of the next pattern in the sequence.

The patterns in the pattern palette were generated such that the correlation coefficient (i.e., a measure of their similarity) between any two patterns in the palette was less than 0.5. This was to ensure that any two patterns were easy to differentiate. For both studies, we used the number of data points in each pattern as a factor, pattern length (L). For each pattern, the difference in any two consecutive data points was randomly selected from a set of five values $\{-2, -1, 0, 1, 2\}$. This was to avoid sudden peaks in the data set.

8.6 Procedure

Participants performed the two studies back to back (typical duration was 60 minutes in total); the order in which participants performed the studies was therefore balanced to counteract learning effects. Each study was divided into two blocks based on technique. At the start of each block, participants were given instructions about the task and the technique used in the block. Before moving to the experimental trials, participants were asked to perform training exercises to ensure that they completely understood the task and the technique. Participants were allowed to perform as many training exercises as they wanted.

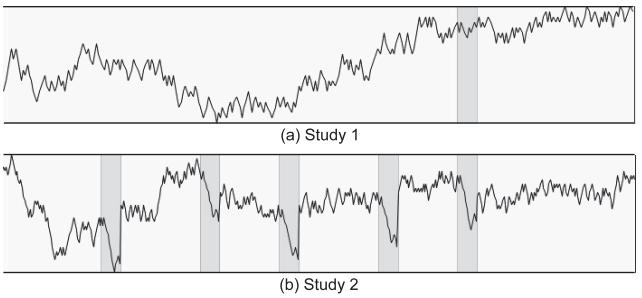


Fig. 8. Example scenarios for both studies.

All trials were interleaved with an intermission screen. Trial time was measured from when the participant clicked on a button to proceed, and until the participant recorded an answer for the trial. We also recorded the correctness for each trial, but this information was not conveyed to the participant. Participants were instructed to try their best to give a correct answer in a minimum amount of time.

9 Study 1: Unguided Visual Search

Searching for recurring patterns is a common task while exploring time-series data. For such tasks, it is common not to have any prior knowledge about the position of similar patterns, hence an unguided search. This was the task we chose for Study 1.

9.1 Hypotheses

H1. *S will yield better correctness than O.* Seeing multiple parts of line graphs simultaneously will let participants search them more accurately.

H2. *S will be faster than O.* For the same reason as H1, stack zooming will help participants finish faster.

9.2 Task

Upon starting a trial, a randomly selected pattern in the data set was chosen as the *template pattern* and was highlighted using a gray region overlaid on top of the line graph (Fig. 8a). The participant was then asked to find the single reoccurrence of this template pattern anywhere else on the graph. We replaced another pattern with the selected pattern to ensure that there was always one identical pattern. Further, the template pattern was drawn in red inside a focus.

Participants were asked to try their best to find an identical pattern, but if they failed they were told to find the segment of the graph that was most similar. To solve the task with overview+detail, participants could create only one focus region on the line graph, while for stack zooming they could create multiple focus regions. To finish a trial, participants were instructed to double click inside the focus

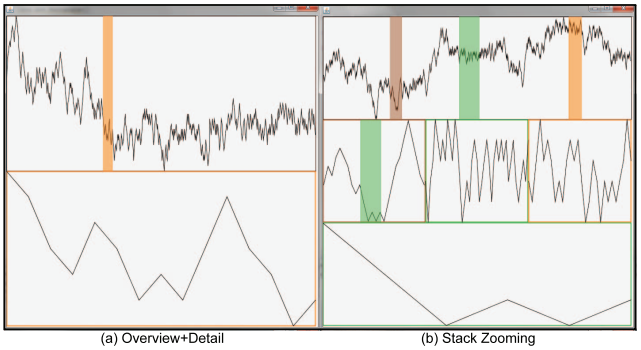


Fig. 9. Techniques T in the user study.

region aligned over the pattern that they wanted to record as their answer. We correlated this pattern with the highlighted input pattern as a measure of correctness. Participants were not allowed to record the starting pattern or any portion of it as their answer.

9.3 Experimental Design

Besides technique T and length L, we also used distance D between template and answer as a factor:

	12	participants
$\times$	2	**Interaction Technique** T (O, S)
$\times$	3	**Pattern Length** L (10, 15, 20)
$\times$	3	**Answer Distance** D (5, 10, 15)
$\times$	2	repetitions
	432	Total trials (36 per participant)

The order of T was counterbalanced and the order of L and D was randomized to avoid learning effects. We measured completion time and the Pearson correlation between the highlighted pattern and the answer. Correlation ranges from -1.0 to 1.0, where 1.0 means the two patterns are exactly same.

9.4 Results

We analyzed the correctness data using a repeated-measures analysis of variance (RM-ANOVA, all assumptions valid) and found a significant effect of interaction technique T on correctness ($F(1,11) = 5.53, p < 0.05$). Fig. 10a shows correlation factor as factor of interaction technique; the average correlation factor for overview+detail was 0.53 (s.d. 0.45) while it was 0.67 (s.d. 0.4) for stack zooming. We found no significant effect of any other factor on correctness.

We also analyzed completion time using an RM-ANOVA. We found that the time measure violated the normality assumptions of the RM-ANOVA, so following common practice, we analyzed the logarithm of the time instead (all assumptions valid). Surprisingly, the analysis showed no significant effect of technique T on completion time ($F(1,11) = 1.33, p = 0.28$). Nevertheless, stack zooming (mean 93 s, s.d. 84 s) performed better than overview+detail (mean 113 s, s.d. 119 s). Fig. 10b shows time by technique.

9.5 Discussion

Our analysis shows that participants achieved better correctness with stack zooming than with overview+detail, confirming our hypothesis *H1*. The reason for this

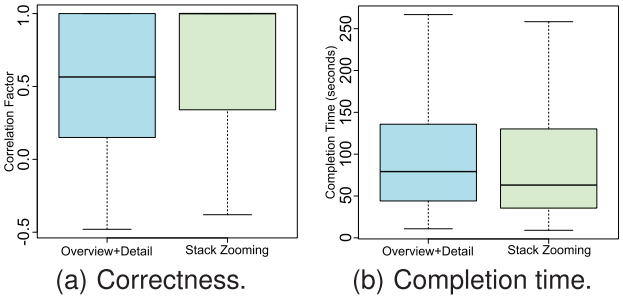


Fig. 10. Performance metrics for Study 1 by T.

difference is likely that being able to create multiple focus regions so that the template pattern was always in view allowed the participants to achieve higher accuracy in their answers.

Stack zooming allowed participants to record their answers quickly while maintaining high accuracy. However, the difference in completion time for the two techniques was not significant, and so we remain unable to confirm our hypothesis *H2*. We speculate that the ability to directly compare their answer with the template pattern encouraged participants to search for more accurate answers and this behavior is reflected in the nonsignificant time difference.

10 Study 2: Guided Comparison

While exploring a time-series data set, it is common to want to compare multiple regions of the data set. Therefore, Study 2 compared stack zooming with standard overview+detail for such a comparison task.

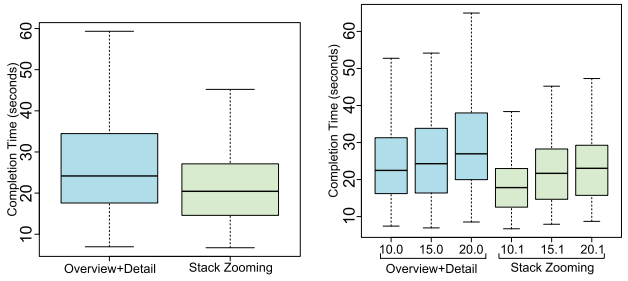
10.1 Hypothesis

H3. *S will perform faster than O.* We think that the multifocus support in stack zooming will help participants compare line graph segments faster than when using standard overview+detail.

10.2 Task

Upon the start of each trial, participants were shown a line graph with multiple rectangular transparent gray regions (Fig. 8b). Each of these rectangular regions was aligned exactly over a randomly selected position in the data set. We modeled the number of rectangular regions as a factor N and restricted it to be an odd number. Furthermore, we generated two additional distinct but very similar patterns (correlation factor between 0.97 and 0.98). We randomly replaced each highlighted pattern with one of the two new patterns, such that across all the N highlighted regions, one pattern is repeated once more than the other.

A task consisted of having the participant finding which of the two new patterns was repeated more often. For standard overview+detail, participants were only allowed to create a single focus region. For stack zooming, on the other hand, they were permitted to create multiple focus regions; if desired, one for each rectangular region. Trials were completed by double clicking inside the focus region aligned over the pattern they thought was dominant. To discourage random answer selection, the system did not



(a) Completion time. (b) Completion time by L.

Fig. 11. Performance metrics for Study 2 by T.

allow the participants to finish a trial without looking at all rectangular regions at least once in a detail window.

10.3 Experimental Design

As in Study 1, we again used technique T and length L as factors, as well as the number of highlights N:

	12	participants
×	2	**Interaction Technique** T (O, S)
×	3	**Pattern Length** L (10, 15, 20)
×	3	**Highlighted Regions** N (3, 5, 7)
×	3	repetitions
	648	Total trials (54 per participant)

The N factor would encourage creating multiple foci, allowing us to measure the effect of space limitations on stack zooming performance. We measured correctness and completion time for each trial.

10.4 Results

Given the nature of the task, the correctness measure for Study 2 was more than 95 percent for both the techniques, and so we did not study this measure further.

We analyzed the completion times using a RM-ANOVA. We again found that the time measure violated the normality assumptions of the analysis of variance, and thus, we used its logarithm in the analysis (all assumptions were met). Fig. 11 shows box plots for completion time as a function of interaction technique T and pattern length L. Significantly, we found an effect of technique T on the completion time ($F(1, 11) = 6.85, p < 0.05$).

10.5 Discussion

Our analysis of the data from Study 2 shows that stack zooming performed significantly faster than standard overview+detail, confirming hypothesis *H3*. This increase in performance is likely a direct effect of the multifocus interaction supported by stack zooming. For the overview+detail technique, participants needed more time because they had to remember the patterns and revisit them time and again to confirm their answer. For stack zooming, on the other hand, they created focus regions for each of the highlighted patterns, and this helped them to perform faster.

11 Conclusions and Future Work

We have presented a theoretical background and a practical implementation of a multifocus interaction technique called *stack zooming* that integrates multiple focus points in skewed-aspect visual spaces with their respective context and relationships. We have also presented four separate instantiations of this idea for different platforms and for different domains, including for stock market data, in collaborative visualization settings, and for asynchronous collaboration on the web. The technique has been validated for log visualization in a case study involving actual network analysts, as well as in a controlled experiment involving 12 human subjects solving comparison and search in time-series data sets using the technique.

Our future work will entail studying the empirical performance of stack zooming in comparison to similar techniques, such as LiveRAC [21], Continuum [24], and Mélange [5]. We also anticipate applying the technique to other domains, as well as studying how to extend it to two-dimensional visual spaces.

Acknowledgments

This work was funded in part by Google, Inc. Conference version previously appeared in IEEE PacificVis 2010 [1].

References

[1] W. Javed and N. Elmqvist, "Stack Zooming for Multi-Focus Interaction in Time-Series Data Visualization," *Proc. IEEE Pacific Visualization Symp.*, pp. 33-40, 2010.
[2] W.S. Cleveland, *Visualizing Data.* Hobart Press, 1993.
[3] B. Shneiderman, "The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations," *Proc. IEEE Symp. Visual Languages*, pp. 336-343, 1996.
[4] G.W. Furnas, "Generalized Fisheye Views," *Proc. ACM SIGCHI Conf. Human Factors in Computer Systems*, pp. 16-23, 1986.
[5] N. Elmqvist, Y. Riche, N. Henry, and J.-D. Fekete, "Mélange: Space Folding for Visual Exploration," *IEEE Trans. Visualization and Computer Graphics*, vol. 16, no. 3, pp. 468-483, May/June 2010.
[6] H. Hochheiser and B. Shneiderman, "Dynamic Query Tools for Time Series Data Sets: Timebox Widgets for Interactive Exploration," *Information Visualization*, vol. 3, no. 1, pp. 1-18, 2004.
[7] S.-M. Chan, L. Xiao, J. Gerth, and P. Hanrahan, "Maintaining Interactivity while Exploring Massive Time Series," *Proc. IEEE Symp. Visual Analytics Science and Technology*, pp. 59-66, 2008.
[8] C. Plaisant, B. Milash, A. Rose, S. Widoff, and B. Shneiderman, "LifeLines: Visualizing Personal Histories," *Proc. ACM SIGCHI Conf. Human Factors in Computing Systems*, pp. 221-227, 1996.
[9] G.W. Furnas and B.B. Bederson, "Space-Scale Diagrams: Understanding Multiscale Interfaces," *Proc. ACM SIGCHI Conf. Human Factors in Computing Systems*, pp. 234-241, 1995.
[10] A. Cockburn, C. Gutwin, and J. Alexander, "Faster Document Navigation with Space-Filling Thumbnails," *Proc. ACM SIGCHI Conf. Human Factors in Computing Systems*, pp. 1-10, 2006.
[11] G. Shoemaker and C. Gutwin, "Supporting Multi-Point Interaction in Visual Workspaces," *Proc. ACM SIGCHI Conf. Human Factors in Computing Systems*, pp. 999-1008, 2007.
[12] C. Gutwin, "Improving Focus Targeting in Interactive Fisheye Views," *Proc. ACM SIGCHI Conf. Human Factors in Computing Systems*, pp. 267-274, 2002.
[13] C. Appert and J.-D. Fekete, "Orthozoom Scroller: 1D Multi-Scale Navigation," *Proc. ACM SIGCHI Conf. Human Factors in Computing Systems*, pp. 21-30, 2006.
[14] T. Igarashi and K. Hinckley, "Speed-Dependent Automatic Zooming for Browsing Large Documents," *Proc. ACM Symp. User Interface Software and Technology*, pp. 139-148, 2000.
[15] 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," *Proc. ACM Symp. User Interface Software and Technology*, pp. 81-91, 1993.

[16] C. Plaisant, D. Carr, and B. Shneiderman, "Image Browsers: Taxonomy and Guidelines for Developers," *IEEE Software,* vol. 12, no. 2, pp. 21-32, Mar. 1995.
[17] K. Hornbæk and E. Frøkjær, "Reading of Electronic Documents: The Usability of Linear, Fisheye, and Overview+Detail Interfaces," *Proc. ACM SIGCHI Conf. Human Factors in Computing Systems,* pp. 293-300, 2001.
[18] H. Richter, J.A. Brotherton, G.D. Abowd, and K.N. Truong, "A Multi-Scale Timeline Slider For Stream Visualization," Technical Report GVU-99-30, GVU Center, Georgia Inst. of Technology, July 2006.
[19] L. Robert and E. Lecolinet, "Browsing Hyperdocuments with Multiple Focus+Context Views," *Proc. ACM Conf. Hypertext,* pp. 293-294, 1998.
[20] J.J. van Wijk and W.A.A. Nuij, "Smooth and Efficient Zooming and Panning," *Proc. IEEE Symp. Information Visualization,* pp. 15-22, 2003.
[21] P. McLachlan, T. Munzner, E. Koutsofios, and S.C. North, "LiveRAC: Interactive Visual Exploration of System Management Time-Series Data," *Proc. ACM SIGCHI Conf. Human Factors in Computing Systems,* pp. 1483-1492, 2008.
[22] A. Zanella, M.S.T. Carpendale, and M. Rounding, "On the Effects of Viewing Cues in Comprehending Distortions," *Proc. Nordic Conf. Human-Computer Interaction,* pp. 119-128, 2002.
[23] E. Pietriga, O. Bau, and C. Appert, "Representation-Independent in-Place Magnification with Sigma Lenses," *IEEE Trans. Visualization and Computer Graphics,* vol. 16, no. 3, pp. 455-467, May 2010.
[24] P. André, M. Wilson, A. Russell, D.A. Smith, A. Owens, and M.C. Schraefel, "Continuum: Designing Timelines for Hierarchies, Relationships and Scale," *Proc. ACM Symp. User Interface Software and Technology,* pp. 101-110, 2007.
[25] W. Aigner, S. Miksch, W. Müller, H. Schumann, and C. Tominski, "Visual Methods for Analyzing Time-Oriented Data," *IEEE Trans. Visualization and Computer Graphics,* vol. 14, no. 1, pp. 47-60, Jan./Feb. 2008.
[26] J.D. Mackinlay, G.G. Robertson, and S.K. Card, "The Perspective Wall: Detail and Context Smoothly Integrated," *Proc. ACM SIGCHI Conf. Human Factors in Computing Systems,* pp. 173-179, 1991.
[27] T. Saito, H.N. Miyamura, M. Yamamoto, H. Saito, Y. Hoshiya, and T. Kaseda, "Two-Tone Pseudo Coloring: Compact Visualization for One-Dimensional Data," *Proc. IEEE Symp. Information Visualization,* pp. 173-180, 2005.
[28] W. Javed, B. McDonnel, and N. Elmqvist, "Graphical Perception of Multiple Time Series," *IEEE Trans. Visualization and Computers Graphics,* vol. 16, no. 6, pp. 927-934, Nov./Dec. 2010.
[29] B.B. Bederson, J. Grosjean, and J. Meyer, "Toolkit Design for Interactive Structured Graphics," *IEEE Trans. Software Eng.,* vol. 30, no. 8, pp. 535-546, Aug. 2004.
[30] K. Perlin and D. Fox, "Pad: An Alternative Approach to the Computer Interface," *Computer Graphics,* vol. 27, pp. 57-64, 1993.
[31] J. Heer and M. Agrawala, "Design Considerations for Collaborative Visual Analytics," *Information Visualization,* vol. 7, no. 1, pp. 49-62, 2008.
[32] M. Wattenberg, "Baby Names, Visualization, and Social Data Analysis," *Proc. IEEE Symp. Information Visualization,* pp. 1-7, 2005.
[33] J. Heer, F.B. Viégas, and M. Wattenberg, "Voyagers and Voyeurs: Supporting Asynchronous Collaborative Information Visualization," *Proc. ACM SIGCHI Conf. Human Factors in Computing Systems,* pp. 1029-1038, 2007.
[34] F.B. Viégas, M. Wattenberg, F. van Ham, J. Kriss, and M.M. McKeon, "Many Eyes: A Site for Visualization at Internet Scale," *IEEE Trans. Visualization and Computer Graphics,* vol. 13, no. 6, pp. 1121-1128, Nov./Dec. 2007.
[35] M. Bostock, V. Ogievetsky, and J. Heer, "D3: Data-Driven Documents," *IEEE Trans. Visualization and Computer Graphics,* vol. 17, no. 6, pp. 2301-2309, Dec. 2011.
[36] K. Kim, W. Javed, C. Williams, N. Elmqvist, and P. Irani, "Hugin: A Framework for Awareness and Coordination in Mixed-Presence Collaborative Information Visualization," *Proc. ACM Conf. Interactive Tabletops and Surfaces,* pp. 231-240, 2010.
[37] G. Mark, A. Kobsa, and V.M. González, "Do Four Eyes See Better than Two? Collaborative versus Individual Discovery in Data Visualization Systems," *Proc. Int'l Conf. Information Visualization,* pp. 249-255, 2002.
[38] P. Isenberg and S. Carpendale, "Interactive Tree Comparison for Co-Located Collaborative Information Visualization," *IEEE Trans. Visualization and Computer Graphics,* vol. 13, no. 6, pp. 1232-1239, Nov./Dec. 2007.
[39] M. Tobiasz, P. Isenberg, and S. Carpendale, "Lark: Coordinating Co-Located Collaboration with Information Visualization," *IEEE Trans. Visualization and Computer Graphics,* vol. 15, no. 6, pp. 1065-1072, Nov./Dec. 2009.
[40] K. Kim, T.D. Kulkarni, and N. Elmqvist, "Interaction Workspaces: Identity Tracking for Multi-User Collaboration on Camera-Based Multi-Touch Tabletops," *Proc. IEEE VisWeek Workshop Collaborative Visualization on Interactive Surfaces,* 2009.
[41] K. O'Hara and A. Sellen, "A Comparison of Reading Paper and On-Line Documents," *Proc. ACM SIGCHI Conf. Human Factors in Computing Systems,* pp. 335-342, 1997.
[42] D. Byrd, "A Scrollbar-Based Visualization for Document Navigation," *Proc. ACM Conf. Digital Libraries,* pp. 122-129, 1999.

Waqas Javed received the Bachelor of Science in Electrical Engineering degree from the University of Engineering and Technology, Lahore, Pakistan, in 2007. He is currently working toward the PhD degree in the School of Electrical and Computer Engineering at Purdue University, West Lafayette, IN. His research interests include information visualization, visual analytics, and human-computer interaction. He is a student member of the IEEE.

Niklas Elmqvist received the PhD degree from Chalmers University of Technology, Göteborg, Sweden, in 2006. He is an assistant professor in the School of Electrical and Computer Engineering at Purdue University, West Lafayette, IN. He was previously a postdoctoral researcher at INRIA, Paris, France. He is a member of the IEEE and the IEEE Computer Society.