A Layer-Oriented Interface for Visualizing Time-Series Data from Oscilloscopes

Roberto Lopez-Hernandez* — École de technologie supérieure
David Guilmaine† — École de technologie supérieure
Michael J. McGuffin‡ — École de technologie supérieure
Lee Barford§ — Agilent Laboratories

Abstract

We present a prototype interface for visualizing and interacting with univariate time-dependent digital signals, i.e., the kind of signals that might be measured using an oscilloscope and then analyzed to find anomalies in edge timing, overshoot, or other measured characteristics. Our prototype visualizes the data using a standard eye diagram that wraps the signal in time, as well as a parallel coordinates view, and a novel 2.5D layer-based technique based on a drawer metaphor for spreading the eye diagram apart and de-occluding the traces within it. Once a "drawer" is opened, the traces within it can be sorted or grouped according to various attributes, and drilling-down into data can be achieved by opening sub-drawers. We also present a novel technique for browsing eye diagrams called temporal brushing. Initial user feedback from students in electrical engineering is reported. A video demonstrating our prototype can be downloaded at http://profs.logti.etsmtl.ca/mmcguffin/

Index Terms: H.5.2 [Information Interfaces and Presentation]: User Interfaces—Interaction styles; I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques

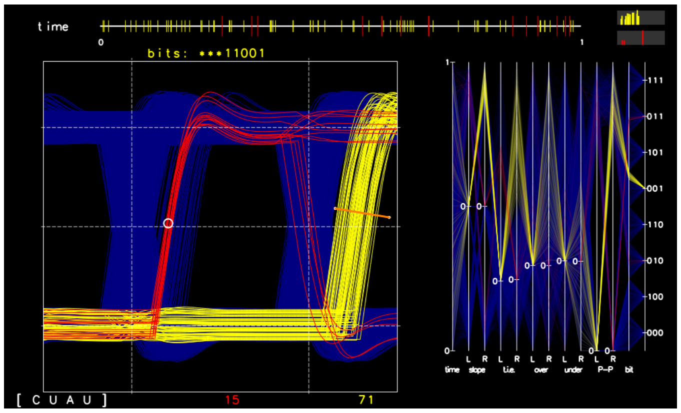


Figure 1: Our prototype interface. At left is an *eye diagram* which shows a signal wrapped around in time as many overlapping *traces*. At right is a *parallel coordinates view* of attributes of the traces. Along the top is a *time-line* of the entire signal. In the top right corner are two *pseudo-spectra* of the highlighted and selected traces. Highlighted and selected traces are shown in red and yellow, respectively, in all views. Immediately above the eye diagram, the label "bits: ***11001" indicates that the currently selected traces have the bit prefix 11001 in common.

1 Introduction

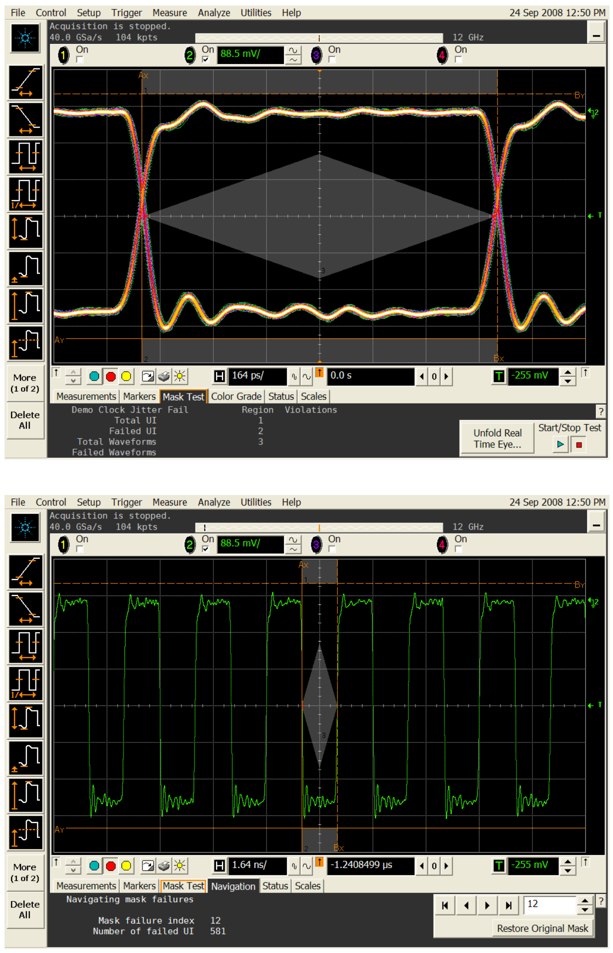
Modern high-end oscilloscopes (such as those in Agilent's Infinium series) can sample signals billions of times per second. Large datasets may be acquired by sampling even over just a small time scale. However, the user interface and visualization capabilities on such instruments are largely based on traditional scope interfaces, updated with a status quo point-and-click mouse-driven GUI. There also remains a strong conceptual model in the interface of a *stream of data* transiently passing through that can be sampled, for example, using triggers, in the hopes of capturing interesting events, rather than capturing a large data set that can then be browsed, visualized, and searched. We feel there are many opportunities to allow the user to interact more directly with the data and to incorporate novel interaction techniques.

This work focuses on visualizing a single digital signal of 1s and 0s. Such signals, as sampled by oscilloscopes, may be thousands to millions of bits long (thus very long in x), but have a very small range in the y direction. One common way to display digital signals is with an *eye diagram*, such as that in the left part of Figure 1. The eye diagram wraps the signal around in time repeatedly, allowing the user to see the range in x of edge transition times, the range in y of 0 and 1 bit levels, and phenomena such as undershoot or overshoot. However, rare events in the signal, such as spikes, edge delays, or other anomalies, may be difficult to see due to occlusion from the rest of the signal. Even when such anomalies are not occluded, and a "mask" may be used to trigger the capturing of such anomalies, the standard interface for browsing such captured events remains primitive (e.g., a dialog box to navigate to "next" or "previous" events, each seen in isolation — see Figure 2).

We investigate a new technique for interacting with eye diagrams by thinking of the many traces within them as individual layers, and allowing these layers to be spread open using a *drawer* metaphor. This allows the user to de-occlude traces and manage the space-versus-overlap tradeoff. Our drawer interface could be described as 2.5D, since the layers always face the camera directly, and are projected orthographically. We also avoid the complexity often incurred by true 3D interfaces by automatically zooming and framing information as the user adjusts the drawer's orientation and length, and automatically snap the drawer to horizontal or vertical angles to facilitate comparison of traces over the x or y axes. Our interface allows the contents of a drawer to be sorted, grouped, selected, and further opened as sub-drawers, to refine the dataset under investigation. Sorting and grouping are performed with a specialized variant of a popup radial menu, allowing users to work in a more gestural fashion, without having to navigate between the data and panels or pull-down menus on the periphery of the screen.

We discuss design issues encountered in developing our drawer-based interaction technique, as well as some complementary techniques inspired by our application domain, including a novel *temporal brush* that allows a user to "unwrap" part of a signal. Initial user feedback from students in electrical engineering indicates that our interface would be useful in certain contexts, and confirms that it allows the user to see the data in ways that are not possible with existing tools. Finally, although our work is applied to eye diagrams of digital signals, our techniques could be used for other time-series

*e-mail: roberto.lopez-hernandez.1@ens.etsmtl.ca
†e-mail: david.guilmaine@gmail.com
‡e-mail: michael.mcguffin@etsmtl.ca
§e-mail: lee.barford@agilent.com


File Control Setup Trigger Measure Analyze Utilities Help
24 Sep 2008 12:50 PM
12 GHz
On
88.5 mV/
Ax
By
Ay
Bx
More (1 of 2)
Delete All
164 ps/
0.0 s
-255 mV
Measurements Markers Mask Test Color Grade Status Scales
Unfold Real Time Eye...
Start/Stop Test
1.64 ns/
-1.2408499 µs
Measurements Markers Mask Test Navigation Status Scales
12
Restore Original Mask

```
Acquisition is stopped.
40.0 GSa/s   104 kpts
```

```
Demo Clock Jitter Fail        Region  Violations
        Total UI                1
       Failed UI                2
     Total Waveforms            3
   Failed Waveforms
```

```
Navigating mask failures

      Mask failure index   12
    Number of failed UI    581
```

Figure 2: Screenshots from the GUI for a commercial oscilloscope. The oscilloscope's GUI is operated with a mouse. *Top*: an eye diagram with a rhombus-shaped "mask" defined by the user. Traces that intersect the rhombus are classified as anomalies, in this case because they transition too late or too early. *Bottom*: the eye diagram is unfolded, and the user may jump forward or backward with buttons in the lower right to view each of the hundreds of anomalies within its local temporal context.

data or x-y plots of data where it makes sense to sometimes wrap along the x direction.

2 Background

An overview of time-series visualizations techniques is found in [1], and recent work includes [13, 16]. Our work differs from much of the previous work by focusing on the special case of a univariate time-dependent signal that varies only within a limited range in y, and that is thus amenable to wrapping in x (producing an eye diagram). Because the signals we seek to visualize have very different ranges in x and y extents, they have extreme aspect ratios when banked to 45 degrees (as recommended by [3]), further motivating the use of wrapping in x. One recently studied interaction technique for time-series visualization is the timebox [11] used to select curves. Timeboxes are useful for removing clutter and focusing on curves of interest; our work shows an alternative way to reduce occlusion that could be combined with timeboxes or other techniques. Horizon graphs [18, 10] are a recent technique for improving space-efficiency by wrapping in y; unfortunately, such wrapping has the side effect of worsening the aspect ratio of an individual signal.

Van Wijk and van Selow [20] propose clustering univariate time series data and displaying the clusters in 2D. They also show a 3D visualization of the time-series data which is similar to the kind of view provided by our drawer-based interface. Their 3D visualization is more or less dismissed in favor of a 2D visualization that allows for clearer comparison of curves. Kincaid and Lam [13] also argue against using 3D, and visualize collections of line graphs that can be sorted or clustered, using color instead of space to encode the dependent variables of time. This eliminates occlusion and allows compact visualizations of large collections to be generated, but "intentionally sacrifice[s] the level of perceivable detail" [13]. Although we share a cautious attitude toward 3D visualizations, we feel that 3D approaches still hold potential, especially when combined with appropriate constraints and interactive techniques that simplify navigation. Our work performs a deeper investigation of design issues around a 3D (or 2.5D) drawer, and clarifies the potential advantages of such an approach.

Outside the realm of time-series visualization, there are examples of work that deal with occlusion of data using lenses [7]. Lenses could be applied to eye diagrams, and can work well if the user knows what they should be filtering for, but otherwise the user may wish for a way to obtain an overview of all the data that is occluded. There are also several examples of work that allow for interaction with layers of information or objects in 2D [15], 2.5D [17], and 3D [5]. We took inspiration particularly from [17], but extended their work, in particular, by allowing sorting and grouping of elements along the z-axis and allowing the user to open sub-drawers.

3 Overview of Our Interface and Approach

Figure 1 shows our prototype interface, written in C++ using OpenGL. The examples in this paper involve synthesized data, due to the difficulty in obtaining real measured data containing anomalies. However, our prototype is also capable of visualizing real data measured with an oscilloscope.

The left part of the window in Figure 1 displays the eye diagram of the signal. To generate the eye diagram, the entire signal is divided into consecutive *traces*, each 2 clock cycles long, and traces are shown overlapping each other in the eye diagram. The time axis in the eye diagram starts and ends midway between clock cycles, so two clock edges (the *left* and *right* edges) are visible in the diagram, showing transitions between 0 and 1 bit levels.

Each trace has several associated attributes. These include the *time* of the trace within the entire signal, and the trace's *bit prefix* (i.e., the 3 bits that make up the trace, plus some number of bits prior to the trace). Additional attributes are computed for each clock edge in the trace, yielding *left* and *right* versions of *overshoot*, *undershoot*, *peak-to-peak* distance, *slope*, and *time interval error* (t.i.e.). The t.i.e. is defined as the distance in x between the trace's edge and the true clock's edge. Traces with anomalous characteristics are of interest to engineers seeking to diagnose problems. For example, traces with a large t.i.e. are more at risk of being misinterpreted during communication. Ideally, the eye diagram should have a large empty space in its center, signifying a small t.i.e. (i.e., the "eye" in the diagram should appear wide and open).

From the perspective of the multiple attributes of these traces, it would seem natural to use a multidimensional visualization technique such as parallel coordinates [12] or a scatter plot matrix [4, 8]. We indeed incorporated a parallel coordinates view into our interface (Figure 1), with which the user can select intervals of traces along any axis. In general, given N dimensions displayed using axes of length L, parallel coordinates have the advantage of only requiring $\Theta(NL^2)$ space[1], whereas a scatter plot matrix would require

[1]This assumes the spacing between axes is proportional to L, to prevent the slopes of line segments from becoming too extreme. If the spacing be-

$\Theta(N^2L^2)$ space[2]. However, both parallel coordinates and scatter plot matrices have the disadvantage of reducing each data element to a single point or polyline. Hence, the shape of a trace over time is not visible with such techniques. As will be seen, however, our drawer technique allows the user to see the shapes of individual traces, and because the drawer can be sorted by any attribute, the user can still "index" into the data set as if it were any of the axes in the parallel coordinates view. Furthermore, with a drawer, the user has continuous control over the tradeoff between how much traces overlap, and how much space is used by the visualization.

Within the eye diagram, the user may select traces by drawing one or more circles or lines. Traces that intersect these are then selected. The user may also select an interval of traces along any axis of the parallel coordinates view. Selections can be built up incrementally by taking either the intersection or the union of subsets of traces chosen within either view.

As the user moves over locations in the eye diagram and parallel coordinates view, prior to selecting anything, the traces under the cursor highlight. This highlighting is coordinated across all four views (eye diagram, parallel coordinates view, time-line, and pseudo-spectra) shown in Figure 1, which can be used for brushing and linking. This can reveal, for example, correlations between t.i.e. and time within the signal, or between other attributes.

4 Layer-Oriented Interaction with a Drawer Metaphor

Once a subset of traces has been selected, the user may open up a drawer containing the selected traces, causing the traces to be spread out. This is shown in several steps in Figure 3. In *A*, we see the eye diagram of the entire data set, and many trace edges are seen to be late with respect to the true clock edges (shows as vertical dotted lines). In *B*, the user has drawn a circle and a line, and traces that intersect both are selected. The selected traces all have a y value of 0 in the middle and 1 after the right clock edge, and many of the traces have a late transition. The arrow-shaped cursor indicates the user may now drag to open a drawer, the result of which is seen in *C*. During opening, the angle and distance that the user drags determines the length and orientation of the drawer. Once opened, the user may "scan" along the drawer by moving their cursor along the drawer's axis. The cursor position is projected onto the drawer's axis, and the corresponding trace's profile is drawn in white with a reference grid (*D*). Hitting a hotkey extends the profile in the x direction, revealing a wider context around the current trace (*E*), that can again be scanned along the drawer's axis. This allows the user to "unwrap" a small portion of the signal. In general, the extended profile and the drawer together provide *two different kinds* of context for the trace that the user points at: one (the drawer) based on whatever attribute is currently used for sorting, and the other (the extend profile) based on the temporal neighborhood around the trace.

By default, the traces in a drawer are sorted by bit prefix. However, in Figure 3*F*, the user decides to sort the traces by time, and does this by invoking a popup radial menu containing all the trace attributes. The user decides to drill down only to traces in the middle of the signal, to avoid transient artifacts that might exist near the start or end of the signal, so they click and drag to select the middle third of the signal (*G*). Next, that subset of traces is collapsed down to a single layer (*H*). At this point, the user could zoom in and once again draw circles or lines to select a smaller subset that could be opened up as a sub-drawer. In this case, however, the user opens the entire collapsed subset as a sub-drawer (*I*). Notice that the axis of the parent drawer is still visible as a white line segment, even though the contents of the parent drawer have been almost completely faded out. The user next decides to sort traces by t.i.e. (*J*), causing traces with a late transition to be moved to the back of the drawer (this can be seen by adjusting the drawer's orientation to create motion depth cues, or by scanning along the axis of the drawer and watching the white profile curve change, or by scanning along the axis and watching the coordinated highlighting in the parallel coordinates view which is not shown here). At this point, the user could decide to *group* the traces into bins according to their t.i.e. (Figure 3*K*: notice the sub-widget in the menu used for grouping). This would cause traces with the largest t.i.e. to be collapsed into the group furthest in the back, and such a group could then be selected and opened up into a sub-sub-drawer.

Instead, however, the user decides to not group traces and instead maintain their sorting by t.i.e., and simply click-drag to select the traces furthest toward the back of the drawer (*L*). That subset, which contains traces of high t.i.e., is then opened up as a sub-sub-drawer (*M*). The user can then scan along the sub-sub-drawer, and may notice that the bit prefix of these traces (displayed along the top of Figure 3*M*) seems to have a common ending. This is confirmed when the user tries grouping by 6 bits (*N*) and then by 5 bits (*O*), the latter resulting in a single group containing all the traces, and having prefix 11001 (*P*). Thus, the late edge transition in this data seems to occur whenever the bit sequence 11001 occurs.

4.1 Automatic Zooming and Framing

Within the eye diagram and drawer, the user may zoom and pan manually at any time. However, the user more typically works with "automatic zooming" turned on, causing the view to automatically adjust as drawers are opened, re-oriented, or closed, so that the data is always centered and framed. This is illustrated in Figure 4.

While a drawer is being opened or re-oriented, if the angle of the drawer's axis comes within a small angle of the horizontal or vertical, the drawer snaps to an axis-aligned orientation, so that its traces are aligned along x or y. This eases comparison. The user may furthermore open a drawer fully so that traces have no overlap between them. A fully opened drawer is equivalent to having tiled viewports, each of which shows one trace (i.e., small multiples [19]), whereas a closed drawer (i.e., an eye diagram) is equivalent to a single viewport showing all traces fully overlapping. These two extremes are commonly found in existing visualization systems, however the drawer also gives the user access to the continuum of intermediate states where traces partially overlap.

4.2 Popup Radial Menus

In the eye diagram, the user switches between modes using a radial menu (Figure 5, upper-left). As with other popup widgets, radial menus have the advantage of only taking up screen space when needed, and of eliminating the need for the user to travel back and forth between their workspace and menus or panels on the periphery. Radial menus have the additional advantage that items within them can be selected faster than in linear menus, encouraging a more gestural kind of interaction [2, 14].

When a drawer is open, the user has access to another radial menu (Figure 5, upper-right, lower-left, lower-right) for sorting and grouping drawer contents. Grouping is done with a specialized sub-widget within which the user drags tangentially, as if they were turning a dial with their mouse. (FlowMenus [9] also allow tangential dragging to control a variable, but do so using different visual feedback.)

4.3 Example Uses

Figures 1 and 3 show the same data set being visualized in our system. In Figure 1, the user selects all traces with a late right edge by drawing a line, and the system immediately displays the common bit prefix of 11001 above the eye diagram. (A common prefix is

tween adjacent axes is kL, $k < 1$, the slope of line segments is limited to $\pm 1/k$, and the entire display will be $(N-1)kL$ units wide and L units high, yielding an area of $k(N-1)L^2 = \Theta(NL^2)$.

[2]Assuming a grid of $(N-1) \times (N-1)$ scatter plots, each $L \times L$ units.

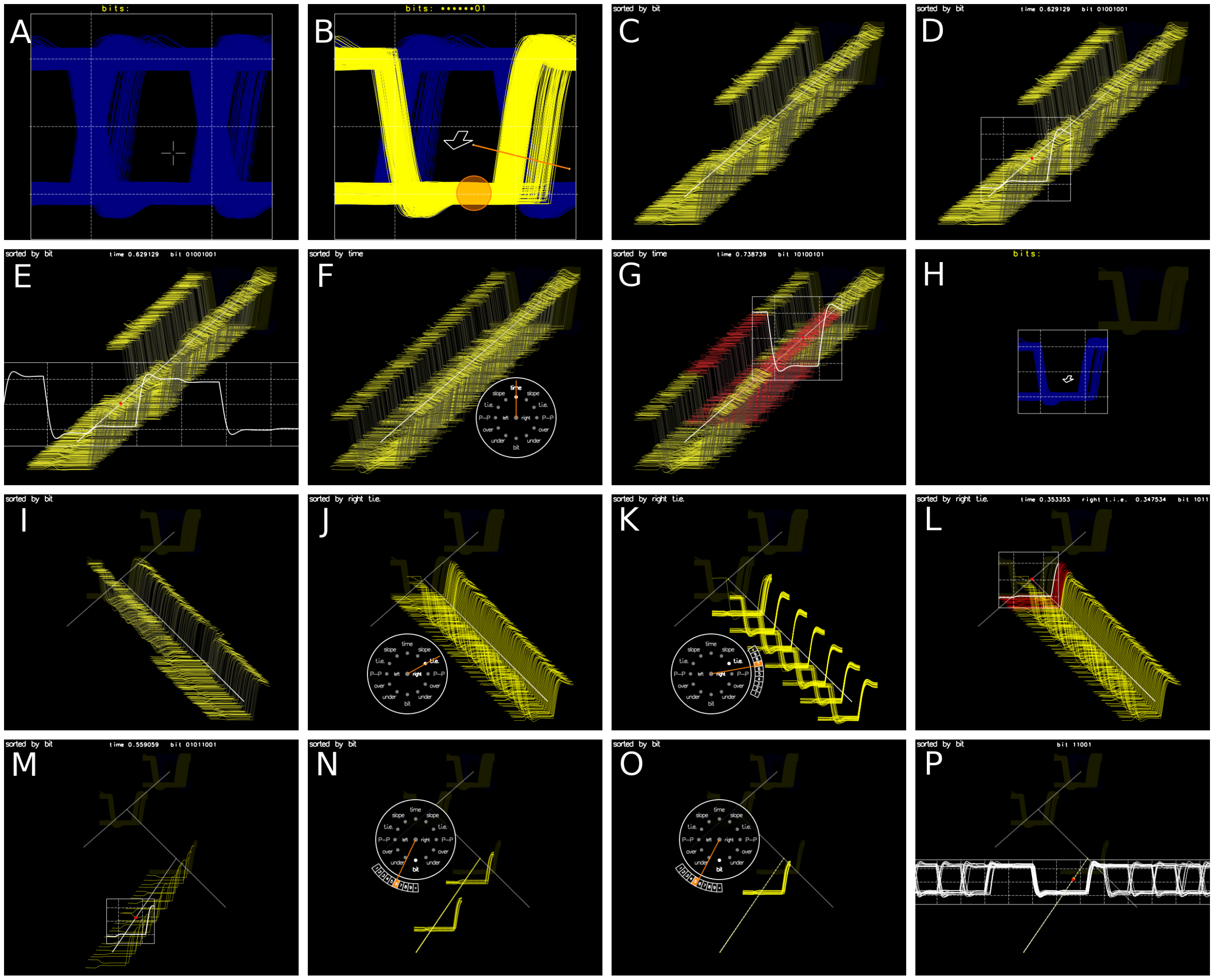


Figure 3: Steps in opening a drawer, sub-drawer, and sub-sub-drawer, to drill down into the data.

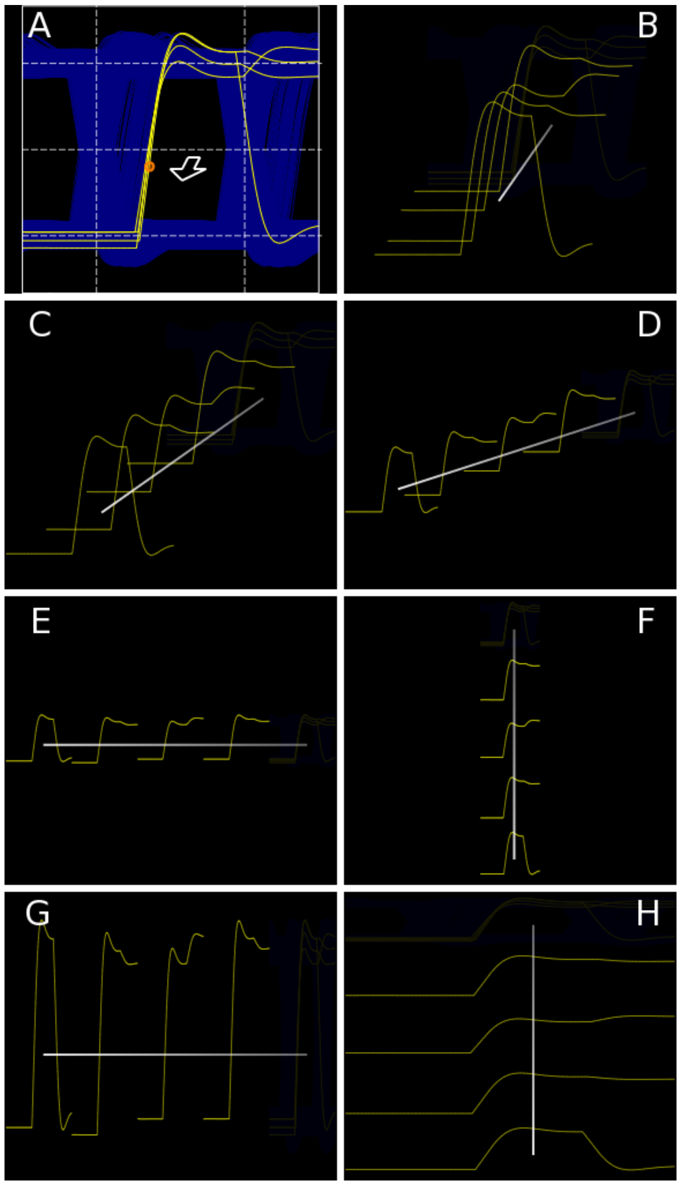


Figure 4: *A*: the user selects a few traces, and enters drawer mode. The arrow-shaped cursor indicates the user may begin opening the drawer. *B-D*: the user opens the drawer by dragging down and to the left. As the spacing between traces increases, the view automatically zooms out. *E-F*: if the user drags out at a nearly horizontal or vertical angle, the drawer snaps to be axis-aligned, to align traces and facilitate comparison. In these cases, the user has also opened the drawer fully, with traces now tiling the space available and no further dragging open is allowed. *G-H*: same as *E-F*, except now *non*-uniform automatic zooming is performed. Traces are scaled independently in x and y to make full use of the space available as the drawer is opened.

also visible in the bit axis of the parallel coordinates display, where we see all yellow line polylines connected to 001. If the user were to adjust this axis to show strings 5 bits long, they would similarly see all yellow polylines connected to 11001.) Figure 3 shows an alternative, richer sequence of actions that arrive at the same discovery using drawers, while along the way allowing the user to see profiles of individual traces and experiment with different sortings and groupings of traces. We also note that in Figure 3, the user

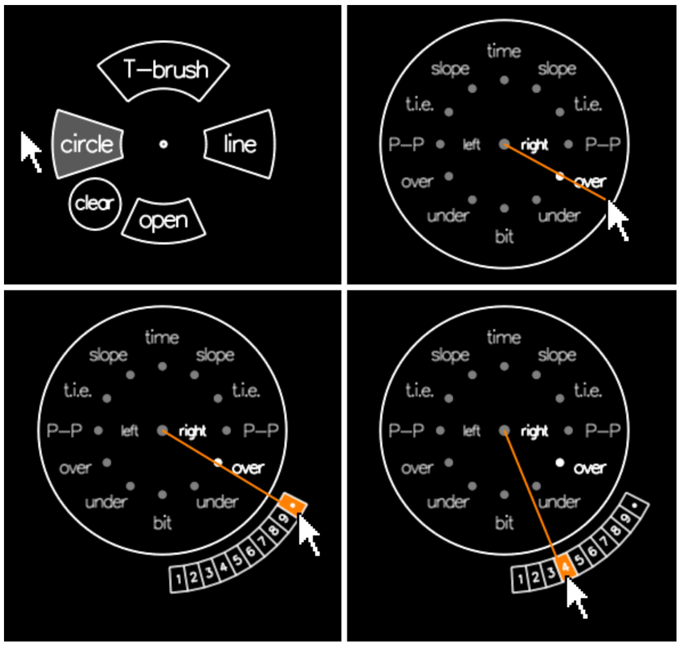


Figure 5: *Upper left*: the radial menu available prior to opening a drawer (or sub-drawer). The user may select *circle* or *line* tools to select traces, *clear* the selection, enter temporal brush (*T-brush*) mode, or enter drawer (*open*) mode. *Upper right*: the radial menu available when a (sub-)drawer is opened. Dragging toward an attribute causes traces in the drawer to be sorted by that attribute. *Lower left*: dragging radially past an attribute opens a submenu of groupings for that attribute, within which the user drags tangentially. *Lower right*: here, the user has selected *4* groupings by overshoot (*over*).

limited their selection to the middle third of the signal. Selecting the middle third of the signal could also be done without drawers by using the parallel coordinates view, however the user would not have the same kind of visual preview that the drawer affords before making such a selection.

Figure 6 shows another data set with an anomaly caused by a specific bit sequence. Again, selecting the anomalous traces can be done with or without a drawer. In this case, however, the anomalies are more challenging to select without a drawer due to them being occluded by other traces. As shown in Figure 6*B*, the user may draw three lines to select the traces of interest, and then see they have a common bit prefix. However, it is unclear if there are any *non*-anomalous traces with the same bit prefix, unless the user also examines the parallel coordinates view. In contrast, by using a drawer, the user can discover that *only* anomalous traces correspond to the particular bit prefix (Figure 6*G-H*), because the drawer de-occludes the traces grouped by bit prefix.

As a final example, Figure 7 shows a waveform that evolves from a sawtooth to a square shape over time. Such a waveform could occur during a transient state in a circuit. The evolution is not easy to see in the eye diagram, although the user could brush over the time axis in the parallel coordinates view to index into the signal and simultaneously watch the eye diagram to see the shape of the waveform change over time. However, this would involve dividing the user's attention between two different views. In contrast, by using the drawer, the user can scan along the drawer axis and watch the white profile curve *in context* as it evolves (Figure 7, bottom).

4.4 Design Issues with Drawers

One issue we encountered in our work was how to best render the contents of a drawer. The first version of our prototype used flat

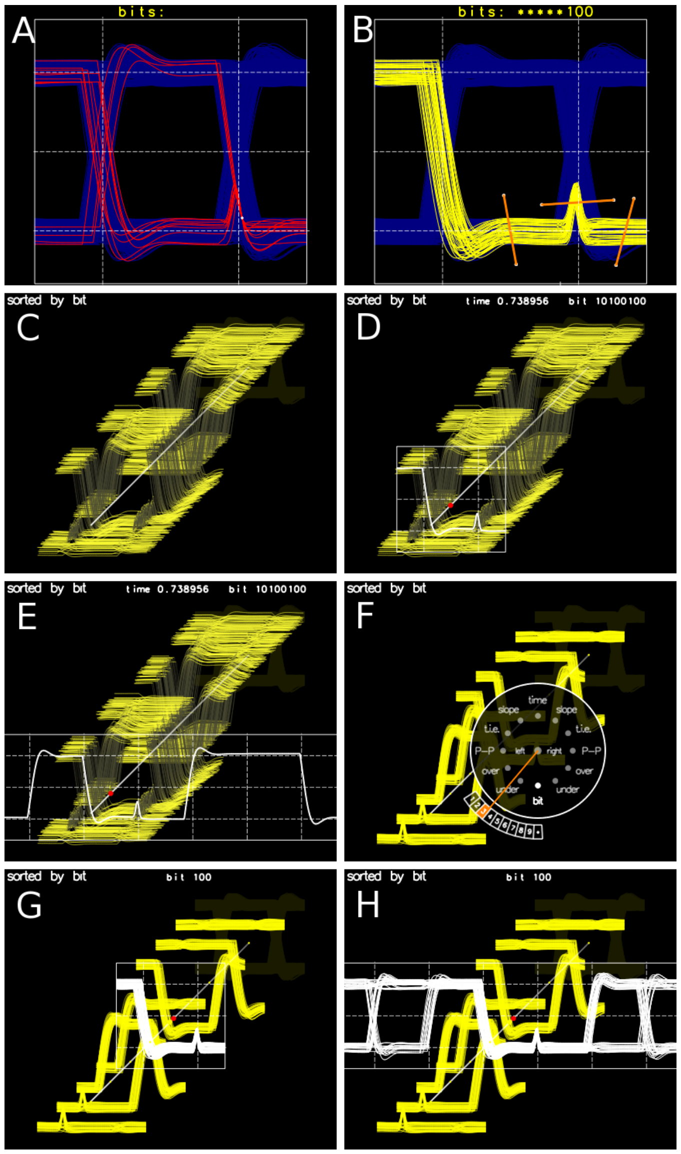


Figure 6: *A*: brushing with the circle tool, the user notices an anomaly in the lower right corner: some traces appear to rise and fall rapidly without transitioning to 1. *B*: one way to select only the anomalous traces is to draw three lines and select the *intersection* of traces crossing all lines. Above the eye diagram, we see the bit prefix 100 is common to all the selected traces. *C-H*: an alternative approach to drill-down to the anomalous traces. *C*: *all* traces are selected and opened in a drawer. *D*: the user scans along the drawer to see the profile of individual traces. *E*: the profile extended in time. *F*: the user groups by 3-bit prefixes (resulting in 8 groups). *G*: traces with the anomaly on the right clock edge are visible only in the group 100, which contains only anomalous traces. *H*: the group's profile extended in time.

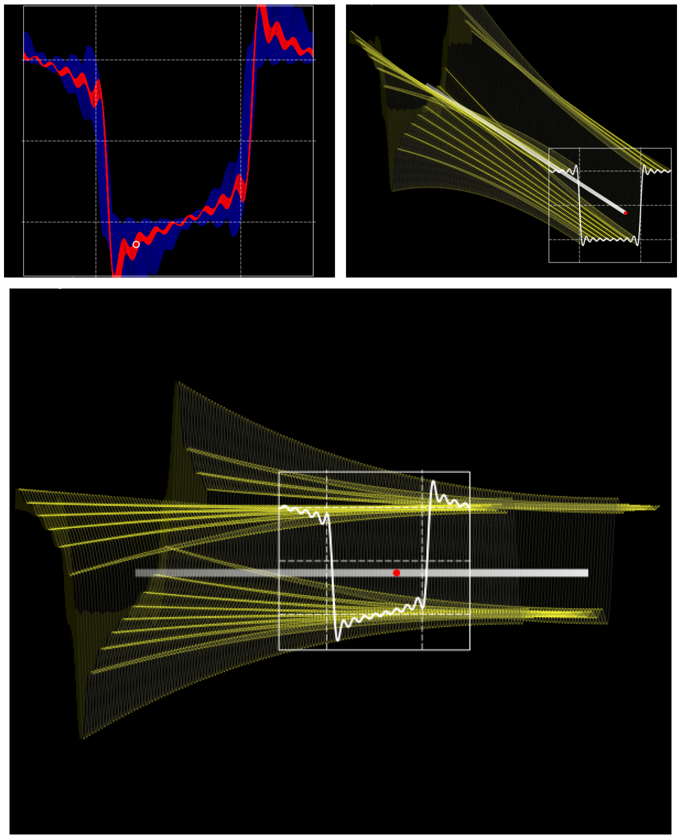


Figure 7: *Upper left*: brushing with the circle tool reveals interesting patterns within the eye diagram, but it is difficult to perceive the evolution over time without switching attention frequently between the time-line (not shown) and the eye diagram while brushing. *Upper right*: selecting all traces and opening a drawer, the user can scan along the drawer and see the profile for any snapshot in time, making the evolution over time more evident. *Bottom*: the drawer snapped to a horizontal angle. It becomes clear that the signal evolves from a sawtooth to a square shape (from left to right, respectively).

shading, drawing all traces with the same color, however this had the disadvantage that when the drawer contained many traces, they would appear as one solid block of color unless the drawer were opened up to be very long. Computing normals and adding lighting often greatly enhanced the distinctiveness of the traces (see, for example, Figure 3*C*-*G*). However, the user can optionally deactivate lighting, which sometimes makes traces clearer (Figure 3*J*-*L*). When the drawer contains a very large number of traces, however, even with lighting it may be difficult to discern individual traces. An additional approach which we have not yet experimented with could automatically downsample the traces shown in the drawer to reduce their density, possibly intelligently choosing key traces whose shape represents a sudden change in the current sorting (possibly taking inspiration from techniques in video volume visualization [6]).

A second issue we considered was how applicable drawer-style interaction would be to other kinds of data, such as layers containing scatter plots, or line segments of parallel coordinate visualizations, or layers of images. We suspect that drawers would not be nearly as effective on such data. In the application we chose, the individual traces are essentially thin curves that often have very similar shapes, and just a small amount of spacing between them can be enough to completely remove the ambiguity caused by overlap. For example, in Figure 4*B*, all four traces are clearly visible despite the overlap between them. If four scatter plots were instead shown with the same degree of overlap, it would be impossible (without using color or motion) to distinguish them. We do, however, suspect that drawer-based interaction could be successfully applied to other kinds of time-series data, especially where the data is limited to a small set of similar shapes such as 0 and 1 bits and transitions between them.

5 Additional User Interface Features

It is possible for a signal to contain an anomaly that is periodic in time. If the anomaly's period is an integer multiple of the signal's clock period, the anomalies will show up in the same location in the eye diagram. If it is not an integer multiple, however, the anomalies will be distributed throughout the eye diagram's time axis (Figure 8). Selecting the traces with such anomalies, however, will cause the traces to show up in the time-line in what appear to be periodic locations. To make it easier for the user to confirm if these locations are truly periodic, we compute pseudo-spectra that are displayed in the upper right corner of the window.

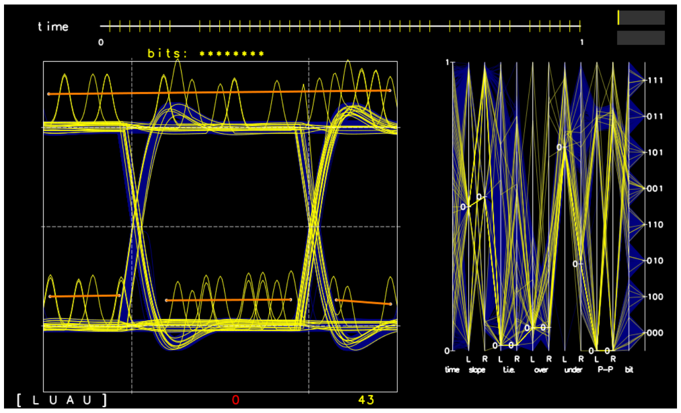


Figure 8: The user draws a few lines (in orange) through anomalies to select the *union* of traces crossing the lines. The time-line shows tick marks corresponding to the selected traces. These tick marks appear to be periodic (except for a few gaps); the periodicity is confirmed by the single yellow spike appearing in the pseudo-spectrum in the upper-right corner.

To compute these, initially we interpreted the timeline to be a vector of 0s and 1s (1s corresponding to highlighted or selected locations) and computed a Fourier transform of this vector. We realized, however, that periodically spaced 1s in the input vector would not result in a single spike in the output. Hence, we implemented an ad hoc algorithm that takes each pair of 1s in the input that are within some maximum distance, and checks to see if other 1s occur at approximately periodic locations. The number of such additional 1s determines the size of the component in the output for that wavelength. Our algorithm results in a single spike appearing in the output if the input contains a periodic pattern of 1s, and two spikes if there are two interleaved patterns of different wavelengths, even when a small number of 1s are missing. Because our method is ad hoc, we call the resulting output *pseudo*-spectra.

Another feature of our interface enables a kind of unwrapping of traces. As shown in Figures 3*E*, 3*P*, 6*E*, 6*H*, a trace's profile can be extended to unwrap a portion of the signal. In that case however, the extent of the unwrapping is limited by the screen width. Figure 9 shows an alternative way of showing the continuation of a trace, backwards and forwards in time, where the extent of the continuation can be adjusted by the user. We call this technique *temporal brushing*. With traditional brushing and linking, corresponding data elements in different views are highlighted together, however with temporal brushing, corresponding (i.e., consecutive) layers are highlighted within the same overlapping space.

6 Initial User Feedback

We showed our prototype to 9 students in electrical engineering at ETS (8 undergraduate students and 1 master's student), all right-handed males, aged 22-29 years, who use computers between 2 and 12 hours per day. 7 of the participants have previous experience

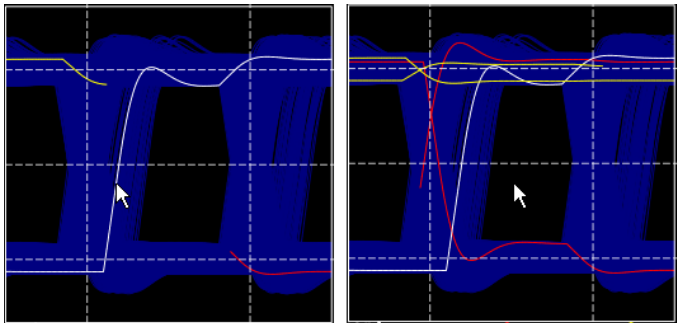


Figure 9: *Left*: the user has selected the *temporal brush* mode. In this mode, yellow and red no longer indicate highlighting or selection. The trace closest to the cursor is shown in white, whose continuation into the past is shown in red, and in the future is shown in yellow. Moving the cursor around updates the display of the trace and its continuation *Right*: The user may also click down and drag to the right or left to increase or decrease, respectively, how far into the past and future they "see". In this case, the user drags right, revealing longer continuations in time.

with 3D user interfaces such as 3D games or 3D modeling software. All have previous experience with oscilloscopes or signal analysis, and 2 have over 100 hours of experience using oscilloscopes.

Each participant was shown the prototype over the course of a 1 hour session, including the time to fill out pre- and post-questionnaires. Participants were asked to "drive" the interface with the mouse and keyboard as features were explained to them and various data sets were loaded for browsing. This gave them time to learn how to operate the interface and to begin speculating how it might be used in real work.

The participants pointed out several minor problems with the prototype that could be addressed without any additional research, such as the ability to change the sizes or locations of circles and lines used for selection, adding undo, mapping the mouse's scroll wheel to zoom, or adding labels to the axes and profile grid.

Reactions to the drawer-based interaction were positive. Comments made in passing by participants upon seeing the drawer or some of its functions include "Excellent [...] Sick!" (Participant 2), and "Fantastic!" (P9). Three participants pointed out that the view shown in Figure 7, bottom, would be useful for showing transient state signals: "You wouldn't necessarily see this without the 3D view" (P1), "[This is] really the positive point ... [This is] a very interesting view [...] With a normal oscilloscope, the view of the signal would be too limited and you would need to scroll a lot [because you can't] overlap [the traces] ... it's really explicit" (P3), "With the 2D view it's impossible to see the evolution of the signal, but with the drawer it's very easy" (P8). Participants suggested improving the drawer by (1) enhancing the extended profile view (such as that in Figure 3*E*) so that users can transition into navigating sideways within it; (2) allowing users to save or bookmark the state of selections and drawers to revisit them later; and (3) allowing users to open multiple sub-drawers from the same drawer, to compare portions of signals side-by-side.

When asked if participants would want our prototype's interface to be available on either an oscilloscope or a PC, 8 out of 9 answered yes (the 9th wrote nothing on the questionnaire for that question). On a scale of 1 (not at all) to 5 (very), participants gave high ratings for the usefulness of the coordinated highlighting (average: 4.5), the drawer (4.6), the extended profile within the drawer (4.4), and the ability to sort or group within the drawer (4.8). The lowest ratings for utility were given to the temporal brush (3.9) and the pseudo-spectra (3.0).

7 Conclusions and Future Directions

We have presented the design of a 2.5D drawer-based interaction technique that gives the user continuous control over the space-overlap tradeoff, enabling them to fully overlap traces (with a closed drawer), or tile traces for side-by-side comparison (with a fully open drawer, assuming the number of traces is small), or access any intermediate state. Our technique allows for a kind of focus + *two* contexts when the user scans along a drawer, where the drawer and the extended profile each provide context for the current trace. Because the drawer can be sorted by any attribute, the drawer can be thought of as behaving like any axis of a parallel coordinates view, where the user can "index into" or "scroll through" the data along any attribute, while also seeing the shape of the traces, without having to divide their attention between the view of the trace and a separate time-line or axis in the parallel coordinates view. Our drawer interaction technique is also novel in allowing sub-drawers for drilling down into data.

We have also presented temporal brushing, a novel technique for unwrapping signals; and presented a radial menu with a specialized sub-menu (Figure 5, lower left and lower right).

Future work could investigate hybrid combinations of drawers and parallel coordinates. For example, the axes in a parallel coordinates view could be enhanced to popup small tooltips that display individual traces, or other visualizations of data, as the user brushes and selects along the axes. Another hybrid which might be useful could combine features of [13] and our system, allowing the user to intermix the space-saving effects of color-encoding and/or drawers of overlapping line graphs, along with detailed focal views of traces of interest.

Additionally, a variant on our drawer technique could space the traces out according to the value of their attribute, rather than space them out evenly as we did, making clusters of values apparent. It could also be interesting to design a variant of the temporal brush applied to drawers, e.g., highlighting consecutive layers in the drawer to show how time is interleaved with the attribute currently used for sorting.

Acknowledgements

We thank Kathy Graham for helping to collect feedback from users at Agilent, Christophe Viau who gave us some visual design tips, Jean-Marie Vilain and Mark Chapuis of Agilent Technologies Canada for providing us with sample data and a demonstration of a commercial oscilloscope, Louis Dessaint for helping us recruit undergraduate users, our participants for their time, and our reviewers for pointing us to relevant literature. This work was funded by Agilent Technologies.

References

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

[2] J. Callahan, D. Hopkins, M. Weiser, and B. Shneiderman. An empirical comparison of pie vs. linear menus. In *Proc. ACM Conference on Human Factors in Computing Systems (CHI)*, pages 95–100, 1988.

[3] W. S. Cleveland. *Visualizing Data*. Hobart Press, 1993.

[4] W. S. Cleveland and M. E. McGill, editors. *Dynamic Graphics for Statistics*. Wadsworth, 1988.

[5] C. Collins and S. Carpendale. VisLink: revealing relationships amongst visualizations. In *Proceedings of IEEE Conference on Information Visualization (InfoVis)*, pages 1192–1199, 2007.

[6] G. Daniel and M. Chen. Video visualization. In *Proceedings of IEEE Visualization (VIS)*, pages 409–416, 2003.

[7] G. Ellis and A. Dix. Enabling automatic clutter reduction in parallel coordinate plots. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 12(5):717–723, 2006.

[8] N. Elmqvist, P. Dragicevic, and J.-D. Fekete. Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation. In *Proceedings of IEEE Symposium on Information Visualization (InfoVis)*, 2008.

[9] F. Guimbretière and T. Winograd. FlowMenu: Combining command, text, and data entry. In *Proceedings of ACM Symposium on User Interface Software and Technology (UIST)*, pages 213–216, 2000.

[10] J. Heer, N. Kong, and M. Agrawala. Sizing the horizon: The effects of chart size and layering on the graphical perception of time series visualizations. In *Proceedings of ACM Conference on Human Factors in Computing Systems (CHI)*, pages 1303–1312, 2009.

[11] H. Hochheiser and B. Shneiderman. A dynamic query interface for finding patterns in time series data. In *Extended abstracts of ACM Conference on Human Factors in Computing Systems (CHI)*, pages 522–523, 2002.

[12] A. Inselberg. The plane with parallel coordinates. *The Visual Computer*, 1(2):69–91, August 1985.

[13] R. Kincaid and H. Lam. Line graph explorer: Scalable display of line graphs using focus+context. In *Proceedings of Advanced Visual Interfaces (AVI)*, pages 404–411, 2006.

[14] G. Kurtenbach and W. Buxton. The limits of expert performance using hierarchic marking menus. In *Proceedings of ACM Conference on Human Factors in Computing Systems (CHI)*, pages 482–487, 1993.

[15] R. Mander, G. Salomon, and Y. Y. Wong. A 'pile' metaphor for supporting casual organization of information. In *Proceedings of ACM Conference on Human Factors in Computing Systems (CHI)*, pages 627–634, 1992.

[16] A. J. Pretorius and J. J. van Wijk. Multiple views on system traces. In *Proceedings of IEEE Pacific Visualization Symposium (PacificVis)*, pages 95–102, 2008.

[17] G. Ramos, G. Robertson, M. Czerwinski, D. Tan, P. Baudisch, K. Hinckley, and M. Agrawala. Tumble! Splat! Helping users access and manipulate occluded content in 2D drawings. In *Proceedings of Advanced Visual Interfaces (AVI)*, 2006.

[18] T. Saito, H. N. Miyamura, M. Yamamoto, H. Saito, Y. Hoshiya, and T. Kaseda. Two-tone pseudo coloring: Compact visualization for one-dimensional data. In *Proceedings of IEEE Symposium on Information Visualization (InfoVis)*, pages 173–180, 2005.

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

[20] J. J. van Wijk and E. R. van Selow. Cluster and calendar based visualization of time series data. In *Proceedings of IEEE Symposium on Information Visualization (InfoVis)*, 1999.