VectorLens: Angular Selection of Curves within 2D Dense Visualizations

Maxime Dumas, Michael J. McGuffin, and Patrick Chassé

Abstract—We investigate the selection of curves within a 2D visualization by specifying their angle or slope. Such angular selection has applications in parallel coordinates, time series visualizations, spatio-temporal movement data, etc. Our interaction technique specifies a region of interest in the visualization (with a position and diameter), a direction, and an angular tolerance, all with a single drag. We experimentally compared this angular selection technique with other techniques for selecting curves, and found that angular selection resulted in a higher number of trials that were successful on the first attempt and fewer incorrectly selected curves, and was also subjectively preferred by participants. We then present the design of a popup lens widget, called the VectorLens, that allows for easy angular selection and also allows the user to perform additional filtering operations based on type of curve. Multiple VectorLens widgets can also be instantiated to combine the results of their filtering operations with boolean operators.

Index Terms—Information Visualization, Finance Visualization, Interaction Technique, Selection Technique, Curves Selection

1 Introduction

Densely overlapping curves, trajectories, and edges occur in many visualizations, including parallel coordinates, time series data, GPS data showing movements of people or vehicles over time, node-link diagrams of graphs, etc. A fundamental task to support in such visualizations is the selection of subsets of curves. Currently, this is possible with a variety of techniques: clicking directly on curves, dragging out a region (by dragging a brush or by drawing a rectangle) that overlaps the curves to select, selecting curves according to some criterion such as "category" or "type" of curve, or sketching the shape of curves to select [1], [2].

Another approach which has been less investigated to date is selection by specifying the angle or slope of curves. Such angular selection could be useful in many cases: selecting one of several overlapping clusters within a parallel coordinates plot, or one of several overlapping edge bundles [3], or selecting all the stocks rising or falling at a given rate on a given date, or all the boats moving along similar trajectories through a given region of an ocean. In some of these examples, automatic clustering might help identify groups of related curves, but clustering algorithms typically require tuning to decide how many clusters to find, whereas a user may visually identify very quickly a group of curves following some common direction and wish to select them.

- *Maxime Dumas is with Ecole de Technologie Superieure of Montreal and with Croesus Finansoft.*
  E-mail: maxime.dumas.1@ens.etsmtl.ca
- *Michael J. McGuffin is with Ecole de Technologie Superieure of Montreal.*
  Email: michael.mcguffin@etsmtl.ca
- *Patrick Chassé is with Croesus Finansoft.*
  Email: patrick.chasse@croesus.com

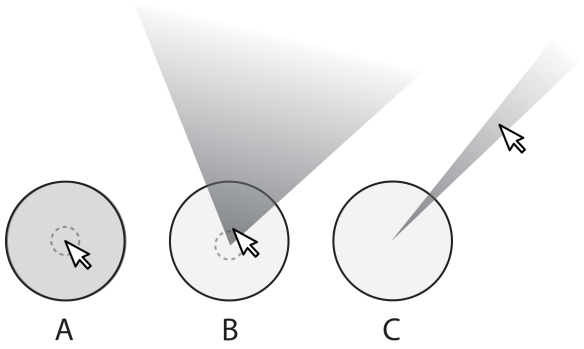

Fig. 1. With angular selection, the user first clicks down to express interest in a circular region (solid black circle in *A*), then drags away from the center by at least the minimum distance shown by the dashed circle (*B*). This selects all curves that pass through the solid black circle and that also have a slope falling within the range of the shaded angular sector. Dragging further away reduces the angular tolerance (*C*). At any time, the mouse wheel can be used to change the diameter of the solid black circle. This technique, by itself, we call the *Pure Angular* technique. We also propose a *Hybrid Angular* technique that supports both angular selection and simple brush selection. In this hybrid technique, the user may optionally release their click within the dashed circle (*A*) to select all curves that pass through the solid circle, regardless of their slope.

In our work, we investigate a technique performed with a single drag for specifying (1) a region of interest within the visualization, (2) an angle or direction, and (3) an angular tolerance (Figure 1). This kind of interaction is well suited for integration into a lens widget that offers the user additional functionality, such as filtering options. As discussed later, our prototype interface allows a

user to instantiate a lens widget, called a VectorLens, and select curves within it based on angular criteria. This is in contrast to previous techniques [4] that don't involve a lens and simply select a range of slopes over some horizontal interval, or other previous work [5] where a user paints multiple regions to define a compound selection of only curves passing through all such regions. Because our approach is oriented toward use with a lens widget, we designed our interaction technique to enable the user to precisely select a subset of curves within the lens with a single click-and-drag, without having to instantiate a second lens (although the user may optionally do so, as presented later). Our approach could sometimes have an advantage over techniques that select slopes over a horizontal interval [4] because the selected curves are limited to a circular region in addition to a limited set of directions or slopes, making it more restrictive.

Before presenting our VectorLens widget, we first study the basic selection technique by which the user selects a region, a direction, and an angular tolerance, by comparing it with other curve selection techniques. Four techniques were experimentally compared: rectangle selection, a circular brush, single-drag angular selection (Figure 1, B and C), and a hybrid combination of circular brush with angular selection (Figure 1, A, B and C). In our experimental task, participants had to select a highlighted target subset of curves by using one of the techniques, and could perform one or multiple interactions (for example, dragging out two or three rectangles in a single trial to select the subset of curves passing through all rectangles). Although the angular and hybrid techniques required slightly more time on average to complete the task, these two techniques also resulted in the largest number of trials that could be completed with a single drag, and also minimized the number of erroneous curves selected during a trial. (This is important for using the technique within a lens widget, as it greatly reduces the need for a user to create multiple lenses for compound selections of curves.) Participants subjectively preferred the hybrid technique overall.

Later, we present the design of the interactive VectorLens widget, which allows the user to perform angular selections in a single drag, as well as brush selections in a single click. Once invoked, the lens remains displayed on the screen, affording further interactions to move the lens, as well as filtering and category-based selection.

Our contributions are (1) the design of a single-drag angular selection technique and of a hybrid technique combining angular selection and a circular brush, (2) an experimental comparison of selection techniques, and (3) the design of the VectorLens widget integrating our hybrid selection technique with extensible filtering and category-based selection features.

2 Related Work

Techniques to aid in the understanding of dense visualizations of points or of curves include excentric labeling [6], lenses that magnify content [7], and lenses that filter out a subset of content [8]. In the case of curves, additional approaches for aiding visualization involve the interactive deformation or "bending" of curves [9].

Techniques for the *selection* of elements in a visualization can be used for invoking operations on the selected elements, and/or for changing rendering parameters (such as the color or alpha) of the selected elements. If the elements to select are small and localized (or point-like), selection of just one element can be aided by techniques that cause the mouse cursor to grow or shrink based on mouse velocity and/or on the proximity of candidate targets near the cursor [10], [11], making it easy for the user to select the nearest target. DynaSpot is one of the most promising of many proposed techniques for aiding such single-target selections. Other single-target selection techniques are notable for their use of direction: Splatter [12], Escape [13], and Click-and-Cross [14] all allow the user to first indicate a location of interest, where there may be multiple overlapping or nearby candidate targets, and then drag in a direction that has been assigned to only one target to complete the selection. In these techniques, the direction is not an intrinsic property of the candidate targets, but is rather assigned arbitrarily to disambiguate them.

Selection of *multiple* points or curves in a visualization can be performed with traditional techniques where the user "paints" a selection using a rectangular or circular brush. Our work focuses on the case where the user wishes to select multiple curves that follow a similar direction and that pass through a common region. Since these target curves may overlap with other curves going in different directions, traditional paint-based selection is ineffective unless the user can paint a *compound* selection of at least two regions, to select the curves passing through both regions. This approach is used with Hochheiser and Shneiderman's [5] timeboxes, where the user can specify two rectangular regions over a visualization of time-series data to select all curves passing through both. Other techniques for selecting curves with a given direction, angle, or slope are angular brushing [4], "angular queries" (another technique proposed by Hochheiser and Shneiderman [5]), and the operators of Guo et al. [15] that have an "angular tolerance". The first two of these apply to parallel coordinates and time series data, but do not naturally apply in the case of parametric curves where neither x nor y is an independent variable, because the techniques assume there is some horizontal interval over which to operate, which would be meaningless in a parametric plot. Also, none of the three previous techniques [4], [5], [15] is optimized to enable the definition of a selection in a single click-drag. In contrast, our present work proposes a technique that can be invoked in a single drag for fast execution, and can be applied to parallel coordinates, time series data, or continuous parametric curves (Figure 3). Our proposed VectorLens widget is also more flexible and complete in its features than these previous widgets.

For example, a single VectorLens widget allows the user to select multiple angular ranges without instantiating additional lenses.

Kosara [16] proposes multitouch input techniques for selecting multiple curves of a given slope or angle using three or four fingers simultaneously. This can work well in some cases, however multitouch input suffers from a lack of spatial precision, as well as physical limits on how far and how close fingers can be positioned.

Various "query-by-sketch" techniques have been proposed [1], [2], [17], [18] to select curves by drawing the approximate shape of the curves of interest. These allow the user to be more specific in their selection than simply specifying a location and a direction, however "query-by-sketch" can also incur more cost to perform, since the user must provide the entire shape of a curve rather than simply a location and direction.

Later in our paper, we show how to incorporate angular selection with a single drag inside a more complete widget called the VectorLens, offering more options to the user. Tominsky et al. [19] recently published a survey of interactive lenses. Among them, an interesting widget for performing selections based on angles is the EdgeAnalyzer [20], which detects clusters of edges, making it easier for the user to select these clusters. However, the EdgeAnalyzer does not allow the user to perform selections with a single drag as our VectorLens does.

In summary, our angular selection technique is unique in that it can be performed in a single drag; and can be applied to parallel coordinates, time series, and visualizations of continuous curves; and is also incorporated in a larger widget so that the user may optionally invoke it in a single drag or invoke the widget's additional options for more control, unlike all previous techniques.

3 Angular Selection in a Single Drag

Our selection technique leverages an elementary and useful characteristic of curves: direction. In many visualizations, curves that pass through similar positions with similar slopes are meaningfully related, e.g., they belong to a "cluster" or "bundle" or somehow represent similar behaviors. Our approach for angular selection allows the user to select such clusters or bundles (or subsets of them) in the absence of automatic clustering algorithms, which would in any case require fine-tuning by the user.

Prior to clicking, a circular brush follows the mouse cursor, remaining centered on the cursor. (This is the solid black circle in Figure 1.) When the user clicks down, the brush is fixed in place. If the user drags outside the dashed circle (Figure 1-B), this defines a vector originating at the brush center and pointing to the mouse cursor position (Figure 2). Only curves passing under the circular brush *and* with slopes oriented in approximately the same direction will be selected. The angular tolerance allowed (i.e., the angle covered by the shaded angular sector) decreases as the cursor is dragged further from the center of the brush (Figure 1). In our implementation, the angular tolerance is initially 70 degrees (when the mouse cursor is immediately outside the dashed circle), and decreases as $1/d$, where d is the distance (Figure 1-C). This inverse function allows a large value initially for coarse selection, but decreases quickly to enable precise selections without having to move too far from the lens center. Theoretically, this technique allows arbitrarily high precision as the distance increases.

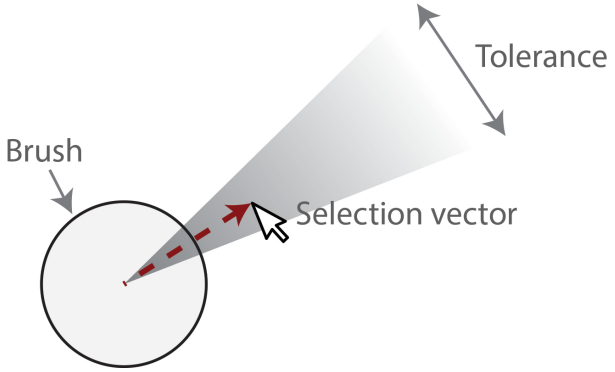

Fig. 2. Elements of our angular selection technique. The brush specifies a region of interest.

Highlighting shows the user a preview of the curves that will be selected prior to releasing the mouse button.

The user may use the mouse wheel to change the diameter of the lens, allowing for the specification of a more or less precise location of interest. Such adjustment can be performed prior to clicking, or even after clicking and during dragging (in which case, the highlighted subset of curves is updated to reflect the lens' new diameter).

The slope of curves under the brush is calculated according to the direction of tangent vectors at the intersections of the curves and the brush circle. If any of the tangent vectors' directions fall within the range of directions of the shaded angular sector, then that tangent vectors' curve will be selected (Figure 3).

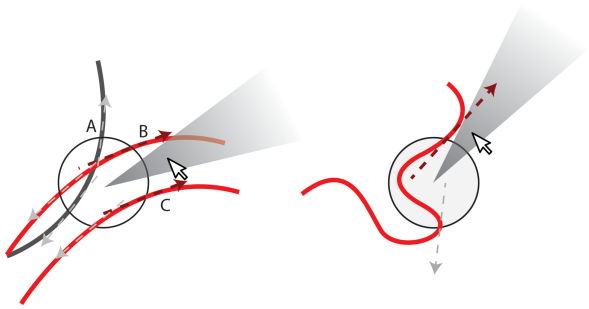

Fig. 3. Tangent vectors are calculated at brush-curve intersection points. Vectors in gray, such as A, are not selected as they do not fit in the angular range. Red vectors such as B and C fit in the tolerance range, causing their curves to be selected.

An alternative approach would have been to compute

the average slope of all tangent vectors at the intersection points, or the average slope between the first and last intersection point, and use this average slope to determine which curves to select. If the curve intersects the circular region at two points, this is equivalent to using the *secant* slope. We did not implement this, because such an average slope can be difficult for a user to predict (as well as being computationally more expensive), and it is easier for the user to drag in the direction of one of the intersection tangents to select their desired curve(s). Furthermore, in practice, the "curves" in parallel coordinates and time series plots usually have long straight-line segments, and would not be as curvaceous as that shown in Figure 3 (right). If a user wanted to select a subset of curves in a parallel coordinates plot, they might first click on an axis and then drag out a range of directions on the right side of the axis, for example, in which case they would not expect their selection to be influenced by the slopes on the left side of the axis, which is another reason to not use the secant or average slope.

Figure 3 shows that curves do not need to be *contained* within the angular sector to be selected, but must have a tangent vector pointing in the same *direction*. The user can often easily avoid unwanted neighboring curves by strategically positioning the brush prior to clicking, so the brush does not cover unwanted curves. In extreme cases where the slope of the curve is highly variable such as Figure 3 right, the user can simply solve this issue by reducing the size of the lens or by leaving the lens centered on top of the curve.

By way of comparison, the "angular queries" proposed by Hochheiser and Shneiderman [5] limit the region of interest only along the 1-dimensional horizontal axis, Our technique allows the region of interest to be positioned within 2-dimensional space.

The angular selection technique just described will be referred to as *Pure Angular* selection, which we evaluated experimentally. We also evaluated a *Hybrid Angular* technique that works just like the Pure Angular technique, except that the user may optionally release the mouse button within the 10-pixel-radius dashed circle in Figure 1-A, causing all curves passing through the brush to be selected. This shortcut allows the technique to be used as a traditional circular brush, which is faster than specifying an angle and is appropriate if there are no undesired curves under the brush.

Notice also the Pure and Hybrid angular selection technique can be applied to parallel coordinates, time series data, edges within network diagrams, and continuous curves.

4 Experimental Evaluation

To evaluate the performance of our angular selection technique, we experimentally compared it with other selection techniques in terms of speed, "errors" (our name for the number of curves whose selection state had to be corrected before the end of a successful trial), and subjective user preferences.

Our angular selection technique was chosen to be easy to integrate into a lens widget, and is designed to allow for selection of an angular subset within a circular region. In practice, this selection can often be done with a single click-and-drag, and with only one circular region defined. However, this click-drag may be somewhat slow to execute because of the precision required on the part of the user to define the angular tolerance. In contrast, other selection techniques (Rectangle and Brush selection, described below) may often require two click-drag actions, but have the advantage that each of these click-drag actions are easy and fast to perform. We chose these techniques for comparison to better understand the tradeoffs between these two styles of interaction: the first based more on a precise selection within a single circular region, and the second based more on fast, multiple selections.

4.1 Task

Participants were asked to select a target cluster within a synthetic data set that was displayed in a seven axis parallel coordinates chart. This is almost equivalent to a discrete time series over 7 time steps, and therefore we do not think a separate evaluation with time series would yield new insights. Four techniques of selection were compared: *Rectangle* selection, where the user drags out the diagonal of a rectangle; *Brush*, where the user clicks to paint with a circular brush (optionally dragging before releasing, to paint a larger selection); *Pure Angular*, shown in Figures 1 B, C; and *Hybrid Angular*, shown in Figures 1 A, B, C, which can be used as a circular brush by clicking and releasing, thus combining the Brush and Pure Angular techniques. Only one technique was available during each trial, but could be invoked multiple times to perform corrective selections or deselections.

With three of the techniques (Brush, Pure Angular, and Hybrid Angular), the user could adjust the size of the brush with the mouse wheel, both before and during a selection.

At the start of each trial, one of the four techniques is active, and the user is shown the full data set across 7 axes, with the target cluster of curves highlighted in red. The user then clicks and drags to activate the technique to attempt to select only the target curves. The successfully selected target curves are then highlighted in green, and the incorrectly selected non-target curves are highlighted in orange (Figure 4, top). If there are also target curves not yet selected, they remain highlighted in red. When the user releases the mouse button, if there are no incorrectly selected curves, the trial ends. Otherwise, the user must perform additional corrective selections (or deselections) with subsequent drags. The trial only ends when all the targets curves have been selected, and no incorrect curves are selected. The *Time* measured is the total duration of the trial. Furthermore,

after each drag, we count the number of non-target curves that are erroneously selected, and the number of target curves that are not yet selected, and the sum of these numbers over all drags of the trial gives the total number of *Errors* for the trial. These are not errors in common sense of the term, because the user is forced to correct these incorrect selections so that there are none left by the end of the trial. However, we are interested in counting such incorrect selections, to compare how close the user can get to the final target selection after just one click-drag, and for simplicity we refer to this count of incorrect selections over the course of the trial as Errors. In practice, techniques with smaller Error count may help the user deal with large datasets, because such techniques cause a smaller subset of curves to be highlighted after the first click-drag, possibly helping the user to more easily identify the subset of curves that interests them, and helping them understand what subsequent corrections (if any) are required.

Corrective selections and deselections can be performed in several ways. Clicking and dragging with the left mouse button adds additional curves to the previous selection (corresponding to a set union of the previous and new selections). Clicking and dragging with the right mouse button removes curves (corresponding to set difference). Holding down the space bar and clicking and dragging with the left mouse button selects *within* the previous selection (corresponding to set intersection). Set intersection is used in Figure 4, top and middle, where a 2nd rectangle is drawn to select only curves passing through both rectangles (in this case, corresponding precisely to the target curves, completing the trial).

Participants were free to use any combination of selections they liked, but were instructed to complete the trials as quickly as possible. We observed during pilots that set intersection with the spacebar was frequently the simplest and most effective approach to use, and encouraged participants to also use this approach when a single drag was insufficient to complete the trial. Usually, 2-3 drags were sufficient to complete a trial using intersection. For example, in Figure 4, top, a single drag with the Rectangle technique cannot select all target curves without also selecting erroneous curves (shown in orange), however a second Rectangle selection with set intersection completes the trial (Figure 4, middle). When using the Hybrid Angular technique, participants were encouraged to perform an initial angular drag for each trial, and if a corrective selection was needed, to do so using the brush shortcut with set intersection.

Participants were also advised to adjust the mouse wheel to obtain a brush diameter they were comfortable with, and then not frequently readjust it, as we found in pilots that frequently readjusting the brush diameter was expensive in time. During the experiment, the selected brush diameter was persistently remembered across trials by the software, reducing the need to readjust it.

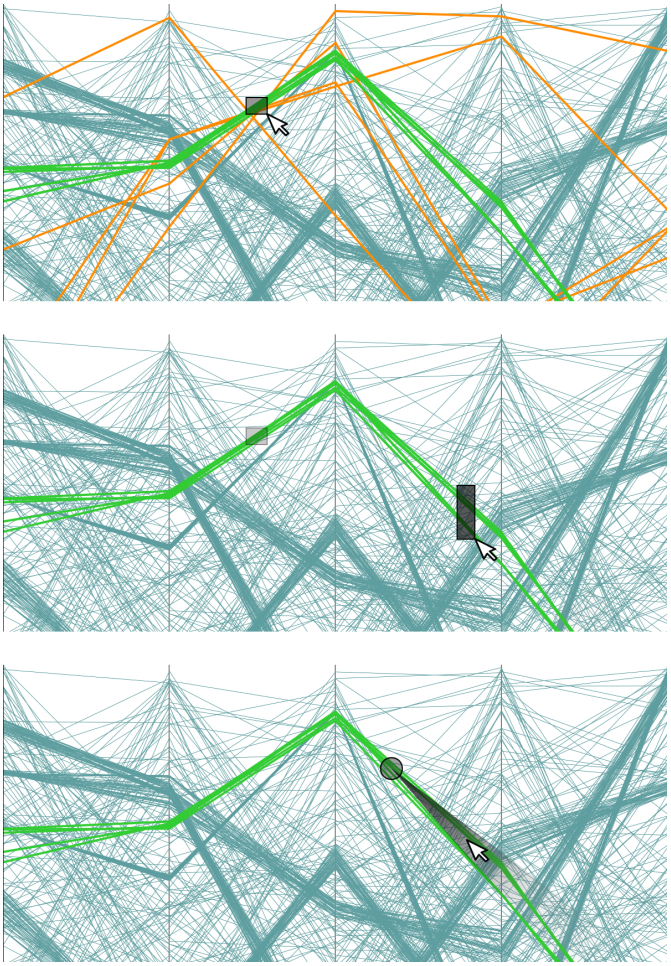

Fig. 4. *Top:* A first drag with the Rectangle technique selects all target curves, but also selects several non-target curves (in orange). *Middle:* a second drag with Rectangle selects all curves passing through both rectangles, which completes the trial. *Bottom:* in this case, the same task can be completed with a single drag using the Pure Angular or Hybrid Angular techniques. Note that the above images are cropped; the experiment displayed parallel coordinates with 7 axes.

4.2 Data Sets

A new data set was randomly generated for each trial, with different random seeds used for each user.

Many solutions have been proposed in the litterature [21], [22] to generate synthetic data sets. We developped our own algorithm to control the density level on the charts. Each data set contained approximately 50% of its curves in 5 clusters, and the other 50% of its curves as uncorrelated background "noise". Within each generated cluster, each curve had to pass within 5% of the axis length of the randomly chosen centroid on each axis for that cluster. The target cluster always consisted of 5 curves, whereas the number of curves in other clusters and in the background depended on the desired *density* of data.

We define density as the fraction of pixels being used

to display curves, taking into account the Euclidean length of line segments, without taking into account overlap between curves. For example, if a single horizontal line of thickness 1 is displayed within a 100×100 window, there are 100 pixels used to display the line out of 10000 pixels in total, yielding a density of 100/10000 = 1%. On the other hand, if the line is diagonal, extending from the lower left corner of the window to the upper right, its length is $100\sqrt{2}$, its thickness is 1, and its total area is $1 \times 100\sqrt{2}$, yielding a density of 1.4%. The density of a visualization with multiple curves is simply the sum of the densities for each curve, without considering overlap. If the total density of a set of curves is 100%, and the curves never overlap, then theoretically all pixels should be covered by curves, however in practice overlapping curves will leave many pixels uncovered even at a nominal 100% density.

In our experiment, we varied the density over 5 levels (Figure 5), from 5% to 90%. Our data set generation algorithm incrementally adds curves until the desired density is reached, resulting in an average of 20 curves for 5% density, up to 380 curves for 90% density. Higher densities result in more overlapping curves, making it more difficult for the non-angular selection techniques to select only the target curves in a single drag.

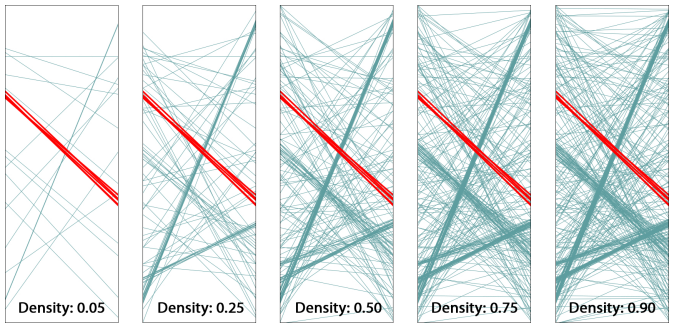

Fig. 5. Cropped images showing the range of densities in the experiment. For all densities, there was a target cluster of 5 curves, initially highlighted in red.

4.3 Apparatus

The experiment was conducted on a 2.3 Ghz Quad Core Intel i7-2820 laptop running Microsoft Windows 7, connected to an external 24 inch 1920×1080 pixel LCD display. Participants used the external display, and external USB mouse and keyboard. Mouse acceleration was disabled.

4.4 Participants

To allow for counterbalancing of conditions and orderings, twelve volunteers (4 women, 8 men) participated, ranging in age from 23 to 46 (average 30.8, median 27.5). All were right-handed and controlled the mouse using their right hand. No one reported having physical handicaps, and none had color deficiencies. Nine participants were employees from Croesus Finansoft working in different departments (programming, QA, business analysis, etc.). Three participants were master's students in engineering programs from ETS. All participants received a twenty dollar gift card for their participation.

4.5 Design

The technique and density variables were within-subjects. Technique was countered-balanced with a 4×4 Latin square. Each user performed 3 blocks of 50 trials with each technique, performing first the 3 blocks for the first technique, then the 3 blocks for the 2nd technique, etc. Pauses were allowed between blocks. The order of densities within each block was random.

Prior to the 3 blocks for each technique, participants were shown a video demonstration of the technique, and then were asked to perform 50 warm-up trials (10 repetitions × 5 densities).

Not counting warm-up trials, there were
12 participants
× 4 techniques (Rectangle, Brush, Pure Angular, Hybrid Angular)
× 3 blocks
× 5 densities (5%, 25%, 50%, 75%, and 90%)
× 10 repetitions
= 7200 trials in total.

Each user session lasted approximately 1.5 hours.

4.6 Results

Figures 6 and 7 show the average time and errors for each technique and density.

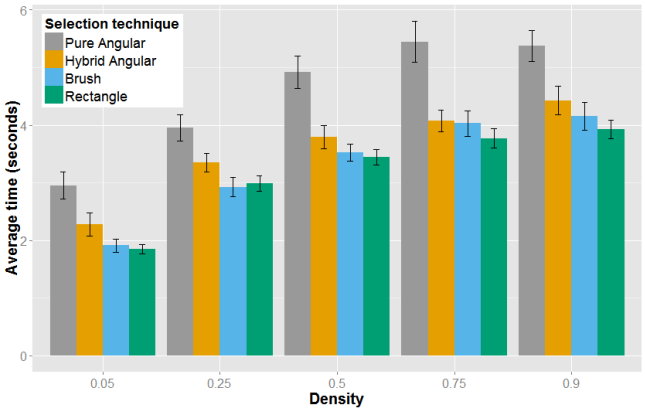

Fig. 6. Average time per density level.

Times were log-transformed and analyzed with a 2-way, 3 technique × 5 density ANOVA with sphericity corrections. Both technique ($F = 29.2$, $p < 0.001$) and density ($F = 384$, $p < 0.001$) had significant effects on time. Pairwise t-test comparisons with Bonferroni correction indicates that all techniques are significantly different except for Rectangle and Brush. Rectangle and

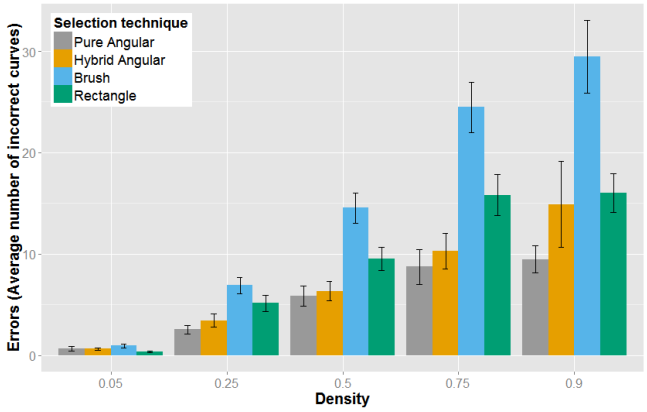

Fig. 7. Average number of erroneously selected curves per density level. Note that all trials ended in success, because users had to correct their selections to end each trial. The "error" count here refers to the number of curves whose selection state was incorrect over the course of the trial and had to be subsequently corrected by the end of the trial.

TABLE 1
Detailed results

Technique	Avg. Time (seconds)	Avg. Errors	Number of single-drag trials
Pure Angular	4.53	5.45	522
Hybrid Angular	3.58	7.12	474
Brush	3.31	15.26	291
Rectangle	3.19	9.37	435

Brush were the overall fastest techniques, with Rectangle slightly (but not significantly) faster.

Errors were also analyzed with a 2-way, 3 technique $\times$ 5 density ANOVA, again finding significant effects due to technique ($F = 25.3$, $p < 0.001$) and due to density ($F = 583$, $p < 0.001$). Pairwise t-test comparisons with Bonferroni correction indicates that all techniques are significantly different, with the two angular techniques the overall best.

We also compared the number of trials in each technique that were successfully completed with a single drag, i.e., completed with zero errors. The two angular techniques performed best by this criterion, as shown in the right-most column of this summary:

All tasks were divided in three blocks of 50 trials. After each block, the participant was allowed to take a pause. We compared the results between blocks and detected no significant difference for selection time and number of errors.

We compared the 4 techniques within each of the 12$\times$5 (participant, density) combinations. We found that Hybrid Angular was the fastest of the four techniques in 8 out of the 60 such combinations. Density does not appear to advantage the hybrid as these 8 fastest times are distributed similarly across all densities. No obvious pattern can be extracted from the data that would explain the ideal conditions for the hybrid technique to outperform the others speedwise.

Participants were observed to exhibit a variety of behaviors and strategies, resulting in varied performance. The fastest participants reported playing video games on a regular basis, with 3 of the top 4 fastest participants playing video games at least 15 hours per week. The two participants who were fastest overall, participants 2 and 9, were also the fastest at using the Hybrid Angular technique. Gaming performance is often influenced by the player speed at executing commands. Experience with advanced selection controls (such as the one found in video games) could have an impact on the observed results. Gamers clearly appeared to be more at ease with the angular controls, and rapidly developped strategies to optimize their selections. This might explain partly why they outperformed the other candidates using the hybrid approach.

After the experimental trials, participants were asked to choose their subjectively preferred technique. 9 out of the 12 participants preferred Hybrid Angular, and the remaining 3 participants preferred Rectangle. The most commonly given reasons for preferring the Hybrid Angular technique was that it allowed for greater "Precision / Limiting the number of selected curves" and allowed for "Rapid correction" of errors.

Because of the Latin square counterbalancing used in our experiment, 3 participants used the Hybrid Angular technique first, before trying the other techniques. All of these participants stated that they would have liked to re-do the trials with the Hybrid technique at the end, feeling that they would have done better a 2nd time.

4.7 Discussion

Of the four techniques, Rectangle is best in terms of time and better than Brush according to all criteria. The angular techniques are best in terms of errors and number of single-drag trials, and between the two, Hybrid Angular achieved a significantly better time than Pure Angular, and Hybrid Angular was only about 12% slower than Rectangle. Hybrid Angular was also the most preferred subjectively.

We thus have a tradeoff between, on the one hand, Rectangle selection, which results in more erroneously selected curves after the first drag (e.g., Figure 4, top) but is nevertheless faster despite the extra time required for corrective selections, and on the other hand, Hybrid Angular selection, which is 12% slower on average but is more precise and was subjectively preferred.

We note again that our metric of "errors" does not refer to unsuccessful trials, since all trials ended successfully, but rather it refers to curves that were at some point in an incorrect selection state during the trial and required correction. Such (temporarily) incorrect selections are often unavoidable for techniques like Rectangle and Brush, and these techniques were nevertheless quite

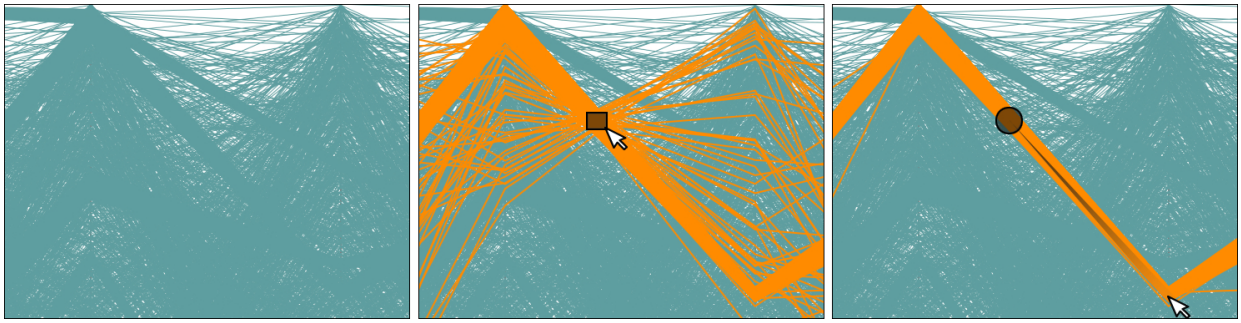

Fig. 8. *Left:* a visualization with a density of 1000%. Some clusters are partially visible, others are entirely hidden. In a realistic scenario, there is typically no highlighting to indicate to the user what to select, and the user must explore to find interesting clusters or subsets. *Middle:* When dragging out a single rectangle, the user cannot avoid selecting many curves that are not part of a single cluster. *Right:* In contrast, a single angular selection allows the user to interactively brush in different directions and focus almost exclusively on a single cluster, all in a single drag.

effective given their time performance, but the lower number of "errors" with the Angular techniques shows that the highlighted subset of selected curves is, on average, closer to the target subset after the first click of the user.

We suspect that one factor penalizing the angular techniques is that participants had extensive prior experience with performing rectangle selection in other software, but had almost no comparative experience with angular selection prior to the experience. A more longitudinal study might find that user performance with angular techniques improves significantly over time. This is also suggested by the wide variety of behaviors and results seen across participants.

Furthermore, in realistic scenarios where target clusters are not highlighted and densities are very high (Figure 8), the user may use these selection techniques to exploratively brush the curves during a drag, without necessarily knowing what direction a cluster has due to occlusion. In such a scenario, rectangle selection could be less effective than angular selection, since highlighting will only appear after clicking, and undesirable curves often cannot be deselected without restarting. In contrast, the use of angular selection allows the direction, tolerance, and circular brush size to all be adjusted during the first drag.

Finally, because angular selection more often allows selection to be completed with a single drag, this makes angular selection more amenable to inclusion within a lens widget with enhanced filtering options, as presented in the next section. A similar widget based on rectangular selection would be more likely to require multiple instantiations to complete a selection, leading to increased screen clutter, especially if the widget contains many auxiliary menus or options.

5 The VectorLens Widget

We propose incorporating angular selection within a lens widget called VectorLens. This widget subsumes the Hybrid Angular technique in our experiment, supporting angular and brush selection each in a single click, as well as supporting direct selection (by clicking on individual curves), categorical filtering capabilities, and complex queries involving multiple lenses. The VectorLens is designed specifically with times series and parallel coordinates in mind. Figures 9-10 show it applied to financial data.

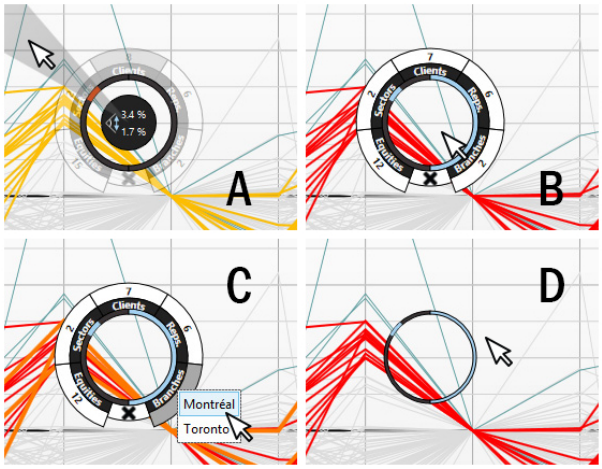

Fig. 9. The VectorLens Widget. *A:* Clicking down and dragging instantiates a VectorLens and invokes angular selection. The innermost ring of the widget highlights in red the angular range currently selected. *B:* Upon releasing, the widget remains on screen, and blue sectors on the innermost ring show the currently selected angular ranges: a full 180 degree range on the right, and a 20 degree range on the left. *C:* Categorical filtering allows the user to further narrow the set of selected curves. *D:* When the cursor leaves the lens, it compresses to take less screenspace, but will return to its original form if the cursor enters again.

When not inside a VectorLens, a dashed circle follows the mouse cursor, to provide a preview of the size of the lens that will be created if the user clicks. This dashed circle is resized with the mouse scroll wheel. Clicking

down instantiates a VectorLens, after which the user may release (to select all curves in the lens, like with Brush selection) or may drag outward and release (to perform angular selection (Figure 9-A)). The user may then access several additional options within the VectorLens which we describe below, or dismiss the lens with the "X" icon at the bottom of the lens, or exit the lens in which case it compresses itself (Figure 9-D). After exiting, the user may instantiate more lenses in other places, or return to the first lens again to access its options.

When the cursor is inside an instantiated VectorLens, the user may click on the innermost ring to redefine an existing angular range, or add a new angular range, either on the left or right half of the ring. The user may also right-click on an angular range on the same innermost ring to remove a range. At any given time, there may be multiple non-overlapping angular ranges defined on the left and right halves of the VectorLens. For example, range R_1 on the left might correspond to curves between 25 and 45 degrees below the horizontal, range R_2 on the left might correspond to curves between 10 and 30 degrees above the horizontal, and range R_3 on the right might correspond to curves between 20 degrees below and 10 degrees above the horizontal. The subset of curves selected in this case could be written as $(R_1 \cup R_2) \cap R_3$. In a time series visualization, the user might have instantiated a VectorLens on a discrete time instant, and in a parallel coordinates plot, the user might have instantiated a lens on a vertical axis. In both cases, the ability to select different ranges on the left and right is useful, since the polygonal curves under the lens will have one slope to the left, and typically a different slope to the right. For example, we might want to select different angles on both sides of an axis (entering in range A AND exiting in range B). Hence, the user may use the left slope, right-slope, or both, to discriminate and select curves. By default, multiple angle ranges on a single side are considered as disjunctions, and selection on both sides would be conjunctions.

Other actions are also possible inside an instantiated VectorLens. The user may click-drag in the center of the lens to move the entire lens to a new location, making it behave like a magic lens [23] for interactive filtering. The user may also position the mouse cursor directly over a curve in the middle of the lens (i.e., within a distance of 1 pixel of the curve) and click to select only that curve. The user may also click on the outermost ring to open a list of categories (Figure 9-C) and select one or more categories in the list, narrowing the selected curves to only those that match the selected categories. This can be done for multiple criteria (displayed in the middle ring: "Equities", "Sectors", "Clients", "Reps", and "Branches" in the financial example in Figures 9-10), in which case the selected curves are narrowed to those matching the chosen categories within all criteria. Note that, prior to selecting within a criterion's list, the user may roll their cursor over the items in the list, to see the corresponding curves highlight (e.g., to see all curves corresponding to a particular Branch, for example). This allows the user to see which curves of each category are present, without having to display a large number of labels on the curves as could be necessary with Excentric labels [6], [24]. The numbers displayed on the outermost ring show the number of categories available under each criterion. The user may also, at any time, use the mouse scroll wheel to change the diameter of the current VectorLens.

When multiple lenses are instantiated (Figure 10), by default an intersection of each lens' selection is computed, i.e., curves are selected if and only if they satisfy the constraints of all lenses. However, the user may modify this behavior using the query builder panel (left side of (Figure 10) to define groups of lenses and to choose the boolean operators to apply.

Normally, each VectorLens will only select among the curves passing through its circular lens area. However, an additional option changes this behavior, causing the widget to select among all curves passing through the horizontal interval covered by the VectorLens (Figure 11). This allows the VectorLens to imitate some previously described widgets, such as [4]. In this "vertical mode", the intersection points used to compute slopes of tangents are the intersections between curves and the vertical boundaries covering the lens' width. This feature gives more flexibility to the tool for special cases where the boundaries of the lens would be too limiting, allowing both constrained and unconstrained modes.

In summary, the VectorLens widget subsumes a more complete set of functions than the previously proposed widgets, while still allowing brush selection and angular selection to be performed each with a single click. It is worth noting that rectangle selection, which was the fastest technique in our evaluation, could also be incorporated into a rich lens-like widget, but this would likely require the user to instantiate more widgets (since selecting the same bundle of curves often requires two rectangle selections, compared with a single angular selection), leading to more screen clutter.

The fluid way in which a user can perform an angular selection and invoke a VectorLens widget in a single click-drag can be compared to other interactive techniques that are not concerned with the selection of data curves. For example, an extensive set of papers have proposed various menu techniques for selecting commands, many of these based on radial widgets [25]–[27]. Many of these are surveyed in [28]. FaST sliders [29] allow a user to perform a rapid parameter adjustment in two drags, and then optionally keep a widget posted to the screen for further interactions, which is comparable to how a VectorLens is available for further interactions after its instantiation. None of these menuing techniques, however, are designed for the selection of elements in a visualization.

6 Conclusion

We have proposed two single-drag selection techniques for dense graphs: Pure Angular, and Hybrid Angular

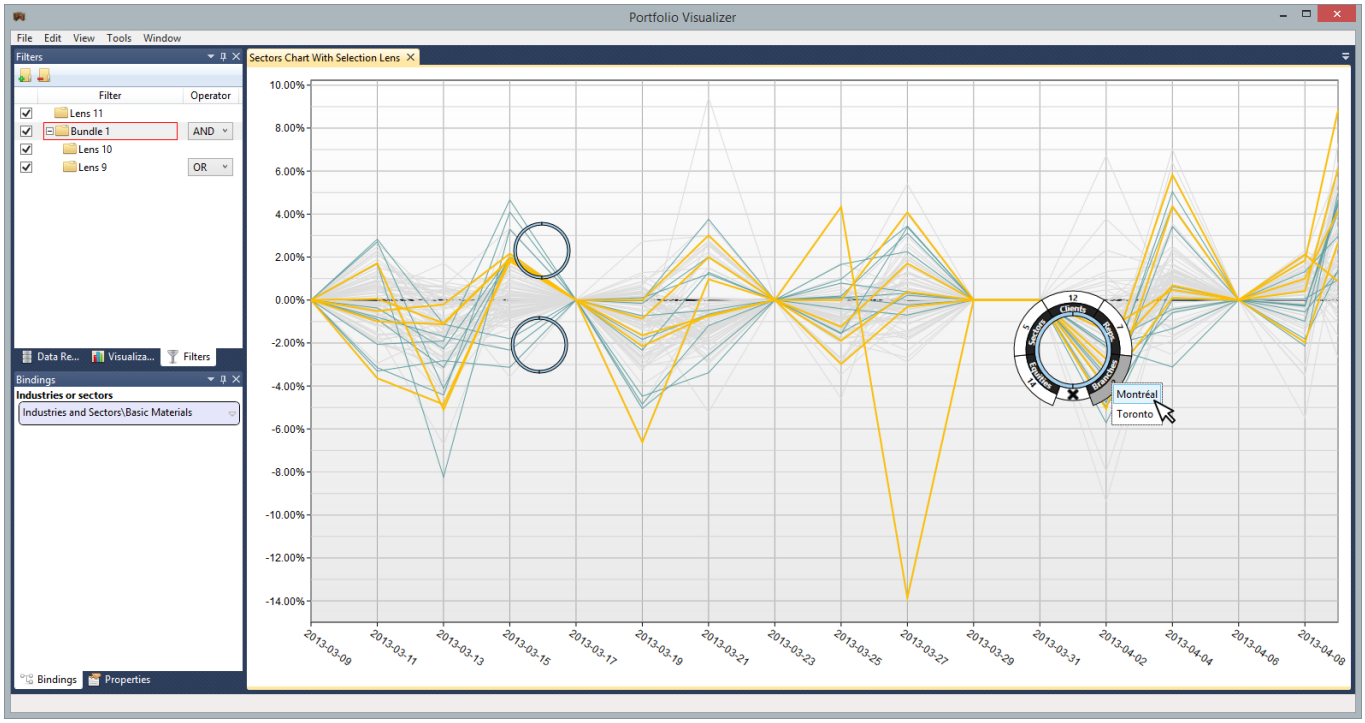

Fig. 10. Complex query created using three lenses. The query builder in the panel on the left shows that the user has grouped lenses 9 and 10 into a single "bundle", and chosen operators corresponding to Lens 11 *AND* (Lens 10 *OR* Lens 9).

(which allows the user to perform angular selection or traditional brush selection). Both of these were compared in a controlled experiment with two other status quo single-drag selection techniques: rectangle selection, and brush selection. Our Hybrid Angular technique was slightly slower than the best of the four, but resulted in significantly fewer errors (i.e., fewer curves whose selection state required correction) over the coures of each trial and more single-drag trials. Hybrid Angular selection was also preferred subjectively by 75% of participants. Finally, we presented the design of VectorLens, a selection widget that subsumes the Hybrid Angular technique without requiring any additional clicks, while also supporting multiple angular ranges, direct selection of single curves, selection based on one or more categories, and complex queries involving multiple lenses, making it more complete than any previously proposed widget for selecting curves.

Our angular selection tool was primarily designed to select clusters or bundles of curves close to each other and oriented in the same direction. Future work could explore automatic clustering techniques to make it easier for the user to "snap" their selection to the nearest cluster-like subset of curves. This could make it easier to eliminate the 2 outlier curves in Figure 8, for example.

Some participants commented during the experiment that our inverse function seemed too sensitive at first when adjusting the angular tolerance. Future work could therefore investigate multiple functions to find a better mapping from cursor distance to angular tolerance, which could improve the performance times with angular selection.

The VectorLens displays categories within simple linear lists. If the number of elements in such a list increased significantly, the user might lose time scrolling. Previous works, such as [30] and [31], provided interesting solutions to accommodate large quantities of data in lists. Such solutions could be applied to the VectorLens to improve scalability to large numbers of categories.

Acknowledgments

Thanks to the participants in our study for their valuable time, and to the members of the HIFIV research group at ÉTS for their feedback. This research was supported by an IIS grant from NSERC, FRQNT, and Croesus, and also by a scholarship from ÉTS.

References

[1] P. Buono and A. L. Simeone, "Interactive shape specification for pattern search in time series," in *Proceedings of ACM Advanced Visual Interfaces (AVI)*. ACM, 2008, pp. 480–481.

[2] C. Holz and S. Feiner, "Relaxed selection techniques for querying time-series graphs," in *Proceedings of ACM Symposium on User Interface Software and Technology (UIST)*. ACM, 2009, pp. 213–222.

[3] D. Holten and J. J. van Wijk, "Force-directed edge bundling for graph visualization," in *Eurographics/IEEE-VGTC Symposium on Visualization (Computer Graphics Forum; Proc. EuroVis 2009)*, 2009, pp. 983–990.

[4] H. Hauser, F. Ledermann, and H. Doleisch, "Angular brushing of extended parallel coordinates," in *Proceedings of IEEE Symposium on Information Visualization (InfoVis)*, 2002, pp. 127–130.

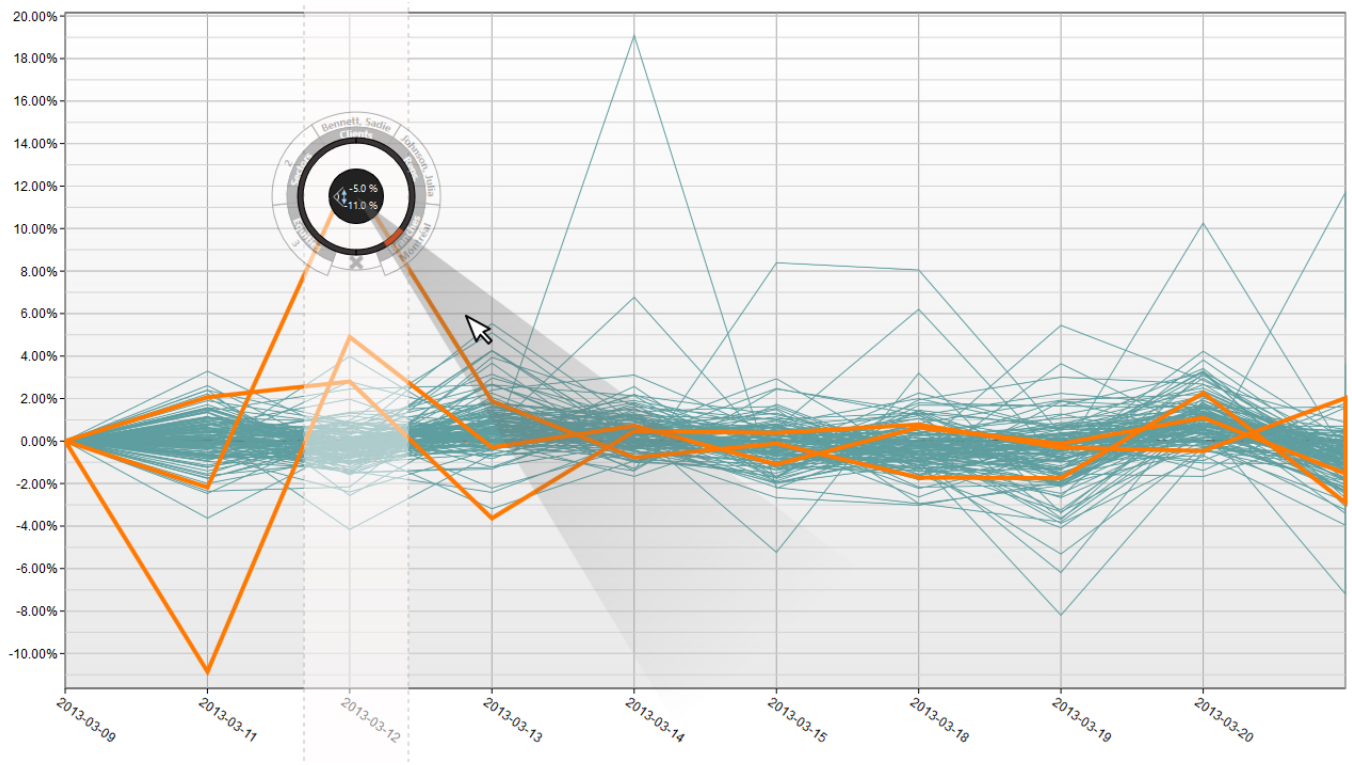

Fig. 11. In this "vertical mode", the horizontal extent of the VectorLens is projected vertically, and all curves in this interval with the appropriate slope are selected.

[5] 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.

[6] J.-D. Fekete and C. Plaisant, "Excentric labeling: dynamic neighborhood labeling for data visualization," in *Proceedings of ACM Conference on Human Factors in Computing Systems (CHI)*. ACM, 1999, pp. 512–519.

[7] E. Pietriga, O. Bau, and C. Appert, "Representation-independent in-place magnification with sigma lenses," *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 16, no. 3, pp. 455–467, Jun. 2010.

[8] G. Ellis and A. Dix, "Enabling automatic clutter reduction in parallel coordinate plots," *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 12, no. 5, pp. 717–724, Oct. 2006.

[9] N. Henry Riche, T. Dwyer, B. Lee, and S. Carpendale, "Exploring the design space of interactive link curvature in network diagrams," in *Proceedings of Advanced Visual Interfaces (AVI)*, 2012, pp. 506–513.

[10] T. Grossman and R. Balakrishnan, "The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area," in *Proceedings of Conference on Human Factors in Computing Systems (CHI)*, ser. CHI '05. New York, NY, USA: ACM, 2005, pp. 281–290. [Online]. Available: http://doi.acm.org/10.1145/1054972.1055012

[11] O. Chapuis, J.-B. Labrune, and E. Pietriga, "DynaSpot: speed-dependent area cursor," in *Proceedings of ACM Conference on Human Factors in Computing Systems (CHI)*. ACM, 2009, pp. 1391–1400.

[12] 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, pp. 428–435.

[13] K. Yatani, K. Partridge, M. Bern, and M. W. Newman, "Escape: a target selection technique using visually-cued gestures," in *Proceedings of ACM Conference on Human Factors in Computing Systems (CHI)*, 2008, pp. 285–294.

[14] L. Findlater, A. Jansen, K. Shinohara, M. Dixon, P. Kamb, J. Rakita, and J. O. Wobbrock, "Enhanced area cursors: reducing fine pointing demands for people with motor impairments," in *Proceedings of ACM Symposium on User Interface Software and Technology (UIST)*, 2010, pp. 153–162.

[15] P. Guo, H. Xiao, Z. Wang, and X. Yuan, "Interactive local clustering operations for high dimensional data in parallel coordinates," in *IEEE Pacific Visualization Symposium (PacificVis)*, 2010, pp. 97–104.

[16] R. Kosara, "Indirect multi-touch interaction for brushing in parallel coordinates," in *Proc. SPIE 7868, Visualization and Data Analysis*, vol. 7868, 2011, pp. 786 809.1–786 809.7.

[17] M. Wattenberg, "Sketching a graph to query a time-series database," in *Proceedings of ACM Conference on Human Factors in Computing Systems (CHI)*, 2001, pp. 381–382.

[18] P. Muigg, J. Kehrer, S. Oeltze, H. Piringer, H. Doleisch, B. Preim, and H. Hauser, "A four-level focus+context approach to interactive visual analysis of temporal features in large scientific data," in *Eurographics/IEEE-VGTC Symposium on Visualization (Computer Graphics Forum); Proc. EuroVis 2008*, 2008, pp. 775–782.

[19] C. Tominski, S. Gladisch, U. Kister, and H. Schumann, "A survey on interactive lenses in visualization," in *Eurographics/IEEE-VGTC Symposium on Visualization (Computer Graphics Forum); State of The Art Report; Proc. EuroVis 2014*, Swansea, United Kingdom, 2014.

[20] A. Panagiotidis, H. Bosch, S. Koch, and T. Ertl, "EdgeAnalyzer: exploratory analysis through advanced edge interaction," in *Proceeding of Hawaii International Conference on System Sciences (HICSS)*, 2011, pp. 1–10.

[21] G. Albuquerque, T. Lowe, and M. A. Magnor, "Synthetic generation of high-dimensional datasets," *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 17, no. 12, pp. 2317–2324, 2011.

[22] S. Bremm, T. von Landesberger, M. He, and D. Fellner, "Pcdc - on the highway to data - a tool for the fast generation of large synthetic data sets," in *International Eurovis Workshop on Visual Analytics; Proc. EuroVA 2012*, Bordeaux, France, 2012.

[23] E. A. Bier, M. C. Stone, K. Pier, W. Buxton, and T. D. DeRose, "Toolglass and magic lenses: The see-through interface," in *Proceedings of ACM SIGGRAPH*, 1993, pp. 73–80.

[24] E. Bertini, M. Rigamonti, and D. Lalanne, "Extended excentric labeling," *Computer Graphics Forum*, pp. 927–934, 2009.

[25] 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)*, 1993, pp. 482–487.

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

[27] S. Pook, E. Lecolinet, G. Vaysseix, and E. Barillot, "Control menus: execution and control in a single interactor," in *Proceedings of ACM Conference on Human Factors in Computing Systems (CHI)*, 2000, pp. 263–264.

[28] G. Bailly, E. Lecolinet, and L. Nigay, "Quinze ans de recherche sur les menus: critères et propriétés des techniques de menus," in *Proceedings of the 19th International Conference of the Association Francophone d'Interaction Homme-Machine*, 2007, pp. 119–126.

[29] M. McGuffin, N. Burtnyk, and G. Kurtenbach, "FaST Sliders: Integrating Marking Menus and the Adjustment of Continuous Values," in *Proceedings of Graphics Interface (GI)*, 2002, pp. 35–41.

[30] Y. Ayatsuka, J. Rekimoto, and S. Matsuoka, "Popup vernier: a tool for sub-pixel-pitch dragging with smooth mode transition," in *Proceedings of ACM Symposium on User Interface Software and Technology (UIST)*. ACM, 1998, pp. 39–48.

[31] B. B. Bederson, "Fisheye menus," in *Proceedings of ACM Symposium on User Interface Software and Technology (UIST)*. ACM, 2000, pp. 217–225.

Maxime Dumas received the master's degree in 2012 from École de Technologie Supérieure of Montréal, Québec, Canada. He is currently working on his Ph.D. degree in the HIFIV group at ÉTS in partnership with Croesus Finansoft. His research interests include finance and security visualization, artificial intelligence and human-computer interactions.

Michael J. McGuffin is an associate professor at ETS, a French-language engineering school in Montreal, Canada, where his students and he do research in information visualization and human-computer interaction. He has published three papers cited more than 100 times each, and in 2009, his paper at the IEEE Information Visualization Conference (InfoVis 2009) received an Honorable Mention.

Patrick Chassé Patrick Chassé is currently director of development at Croesus Finansoft inc in Laval, Québec, Canada. He has a baccalaureate in electrical engineering from École de Technologie Supérieure of Montréal where he graduated in 1994. He worked for several years as a software developper in different areas. He also contributed to several research papers and has participate in several conferences in the technology for persons with disabilities field.