Indirect Multi-Touch Interaction for Brushing in Parallel Coordinates

Robert Kosara

UNC Charlotte, USA

ABSTRACT

Interaction in visualization is often complicated and tedious. Brushing data in a visualization such as parallel coordinates is a central part of the data analysis process, and sets visualization apart from static charts. Modifying a brush, or combining it with another one, usually requires a lot of effort and mode switches, though, slowing down interaction and even discouraging more complex questions.

We propose the use of multi-touch interaction to provide fast and convenient interaction with parallel coordinates. By using a multi-touch trackpad rather than the screen directly, the user's hands do not obscure the visualization during interaction. Using one, two, three, or four fingers, the user can easily and quickly perform complex selections. Being able to change the selections rapidly, the user can explore the data set more easily and effectively, and can focus on the data rather than the interaction.

Keywords: parallel coordinates, interaction, brushing, multi-touch

1. INTRODUCTION

Interaction is a common problem in visualization, in particular when working with large and high-dimensional datasets. A common interaction technique is brushing, which lets the user select a subset of a dataset that is then highlighted. Different views of the same data, or even just the display of these data points in the same view, provide insight into multi-dimensional relationships and structures.

Parallel coordinates[1,2] (Figure 1) are a popular visualization technique for the analysis of high-dimensional, (mostly) numeric data. Interaction in parallel coordinates typically consists of axis reordering and inversion, as well as brushing on one or more axes.[3] There are also extensions of this idea, such as angular brushing,[4] which lets the user brush by direction rather than location, and thus effectively brush different kinds of correlations.

Multi-touch trackpads are becoming more common in laptops such as Apple's *MacBook* and *MacBook Pro* lines. We propose the use of multi-touch interaction on these trackpads for brushing in visualization*. We have developed interaction techniques for axis selection, reordering, single-axis brushing, and two kinds of two-axis brushing. Indirect multi-touch interaction enables the user to quickly explore the data without the effort and attention to the user interface most conventional approaches require.

2. RELATED WORK

Multi-touch interaction has received a lot of attention lately, most of it centered on directly touching the display surface; though there are also some existing indirect approaches. Benko et al.[5] propose several methods that allow for very precise selections, e.g., by using one finger to indicate a scale while another one is used to direct a pointer to the target. Work on indirect (multi-)touch interaction tends to focus on large interaction surfaces[6] and recreating the direct interaction on a different surface when multiple devices are involved.[7] Schmidt et al. showed that such indirect interaction tends to be slower than direct touches,[8] but we believe that especially for visualization, the fact that the user's hands do not obscure the data displayed outweighs this.

Multi-touch interaction has been explored for brushing in information visualization,[9,10] particularly for collaborative work. Like the work described above, this tends to focus on large interfaces and collaboration, and require hardware that is specialized, rare, and expensive.

*A video demonstrating the technique is available at `http://vimeo.com/13437693`. The program can be downloaded from `http://github.com/eagereyes/ParVisMT/downloads`. It requires Mac OS X 10.6 and a MacBook (Pro) to run.

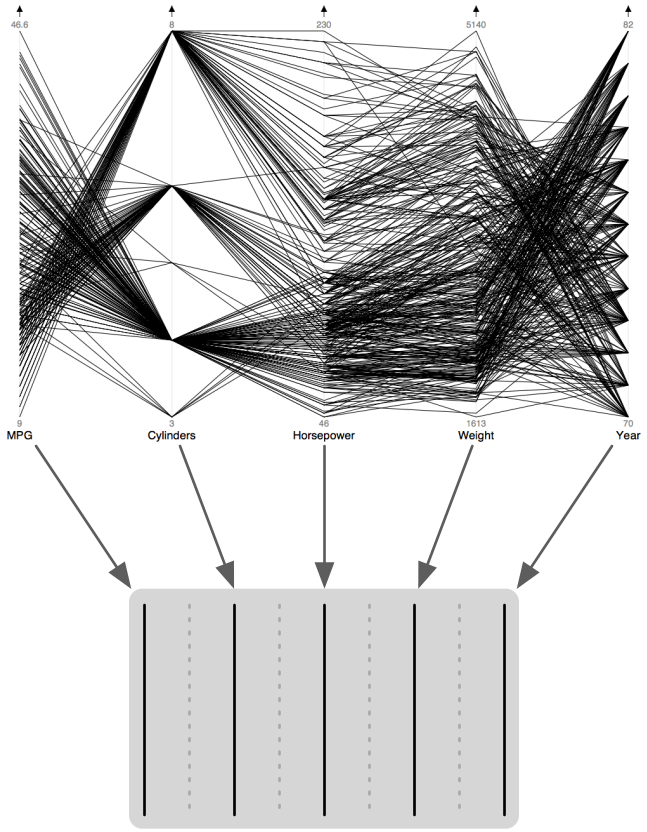


Figure 1: Parallel coordinates showing a dataset about cars (top). Mapping of dimensions to locations on the trackpad, dotted lines indicate the boundaries between axes (bottom).

Our aim is to provide new interactions with minimal requirements for new hardware. Our proposed multi-touch interaction technique can be implemented on commercial hardware and can be incorporated into existing programs quite easily.

3. INDIRECT MULTI-TOUCH INTERACTION

Most multi-touch devices, such as Microsoft's Surface, Han's touch displays,[11] and Apple's iPhone. While these devices provide high resolution interaction and direct manipulation[12] (in an almost literal sense), we propose that indirect multi-touch is superior in some applications.

When touching the display directly, the user's fingers and hands obscure the screen, making it necessary to interrupt the interaction to properly see its results. Resuming the interaction is imprecise and the constant switching from interaction to looking to interaction distracts from the exploration of the data.

Indirect multi-touch interaction solves this problem by placing the multi-touch surface in a different location. Having a small, conveniently located touch surface also increases user comfort and reduces fatigue.

Another benefit of indirect interaction is the use of more abstract interactions. Rather than grabbing and flicking objects, the user can use gestures like swipes to communicate an intent, which can translate into actions that have no physical equivalent (like undo). Visualization involves both relatively literal interactions (like axis reordering and single-axis brushing) and more abstract ones (like angular brushing and axis inversion). By not relying on a direct touching metaphor, both types of interactions can be integrated seamlessly.

4. PARALLEL COORDINATES

Parallel coordinates map the values of each data point onto parallel visual axes, and connect all points belonging to the same data item with line segments.

4.1 Visual Representation of Data

While the initial visual representation in parallel coordinates is not always informative (Figure 1, top), it does provide some information about the data (car models, in our example). We can see that the *Cylinders* and *Year* axes only have a small number of different values (3, 4, 6, and 8 cylinders; 13 years from 1970 to 1982). All other axes have a more continuous distribution of values.

Looking at the lines between axes, one can tell that cars with more cylinders tend to have lower values on the *MPG*, and higher values on the *Horsepower* axis. There also seems to be an inverse relationship between *Year* and *Weight*, meaning that cars made in the later 1970's and early 1980's were lighter than cars made earlier.

4.2 Brushing

To find out more about the data, the user needs to use brushing. Selecting values on one or more axes allows her to find correlations and interesting subsets of the data. In the car models example, brushing the last years (1980–82) would reveal that all the car models introduced in those years were in the lower third of the *Weight* and *Horsepower* axes, and almost all above the middle of the *MPG* axis.

In general, brushing is used to reduce clutter, find data items that match certain criteria (or combinations of criteria), and provide focus and context. Brushing lets the user compare a subset of the data to the entire dataset, and thus develop knowledge about what the data contains.

5. MULTI-TOUCH INTERACTION FOR PARALLEL COORDINATES

The central idea of this work is the mapping of areas on the trackpad to axes on the display (Figure 1). The user can directly pick axes using the horizontal dimension of the trackpad and select a range on an axis using the vertical dimension. The number of fingers used determines the kind of interaction.

5.1 One Finger: Select Axis, Invert

When the user moves a single finger over the trackpad, the axis that corresponds to the finger position is highlighted. This mainly serves as an aid to finding the right axis for brushing, but does not change anything in the visualization. It also helps make the entire interaction model more discoverable: if a single finger causes a reaction, perhaps two will, as well?

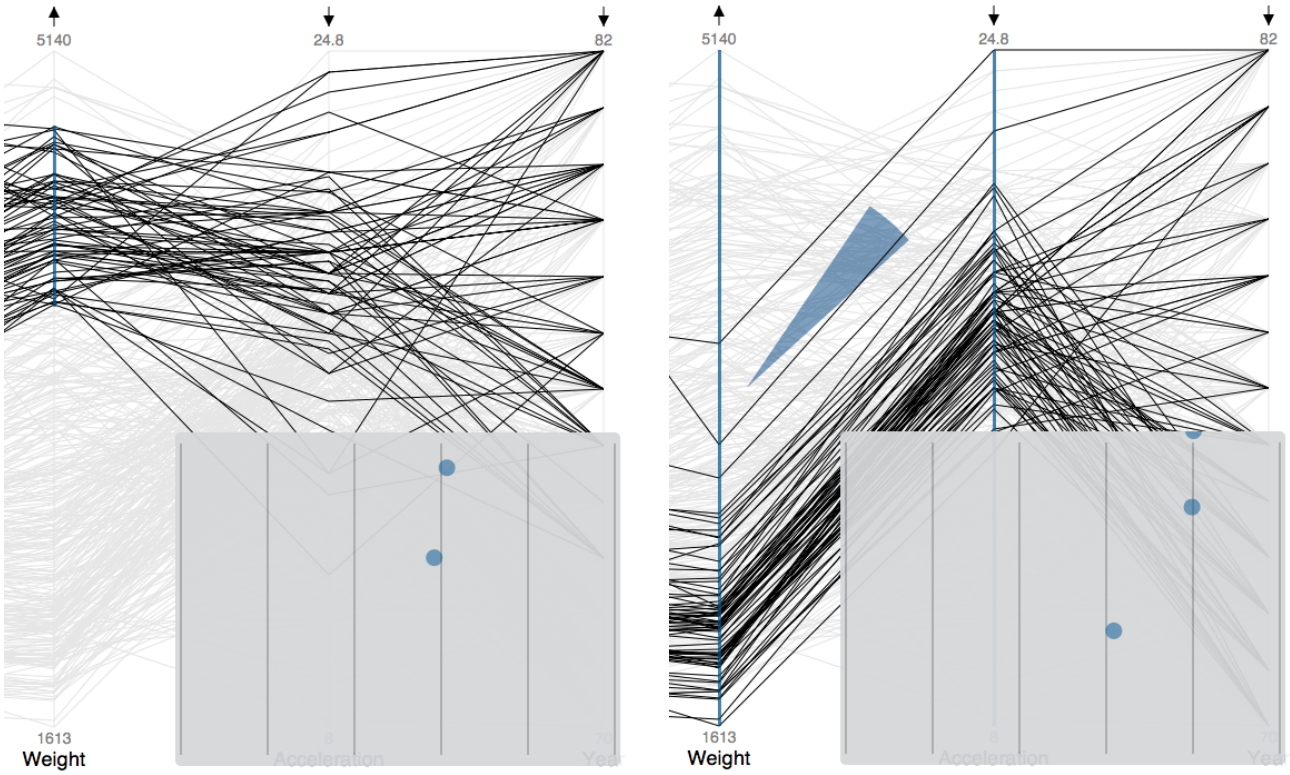
In addition to selecting an axis, a single finger interaction is also used to invert it: double-tapping on an axis turns it upside down. This is useful for testing whether a seeming inverse correlation really exists and to reduce clutter in the display from crossing lines.

5.2 Two Fingers: Brush on One Axis

Two fingers on the trackpad select a range on one axis and brush all the values lying in that range (Figure 2a). If the two fingers are not perfectly aligned vertically, and perhaps even come to sit on different axes, the axis that was previously selected with a single finger is used for brushing. The brush sticks to that axis until the user goes back to one or no fingers on the trackpad (this also persists through three- and four-finger interactions).

The reason for the sticking brush is that during brushing, it is difficult to keep the fingers correctly aligned with the horizontal region the axis occupies on the trackpad. The result would be the brush jumping from axis to axis, which is distracting and disrupts the interaction. It is very easy to indicate the intent to switch to another axis by lifting one finger and selecting a different axis.

Brushing using the mouse typically means selecting a range on an axis, which is then brushed. Some programs interactively show the result of the brush operation during the interaction. Changing the brush means deleting the existing brush or at least replacing it with a new brush interaction that starts from scratch. Being able to quickly and smoothly change the position and size of the brush speeds up selection and requires less attention to the interaction.



(a) Two fingers brushing on a single axis.

(b) Angular brushing on adjacent axes using three fingers.

Figure 2: Multi-touch interaction for brushing in parallel coordinates, two and three fingers. Insets show the positions of fingers on the trackpad.

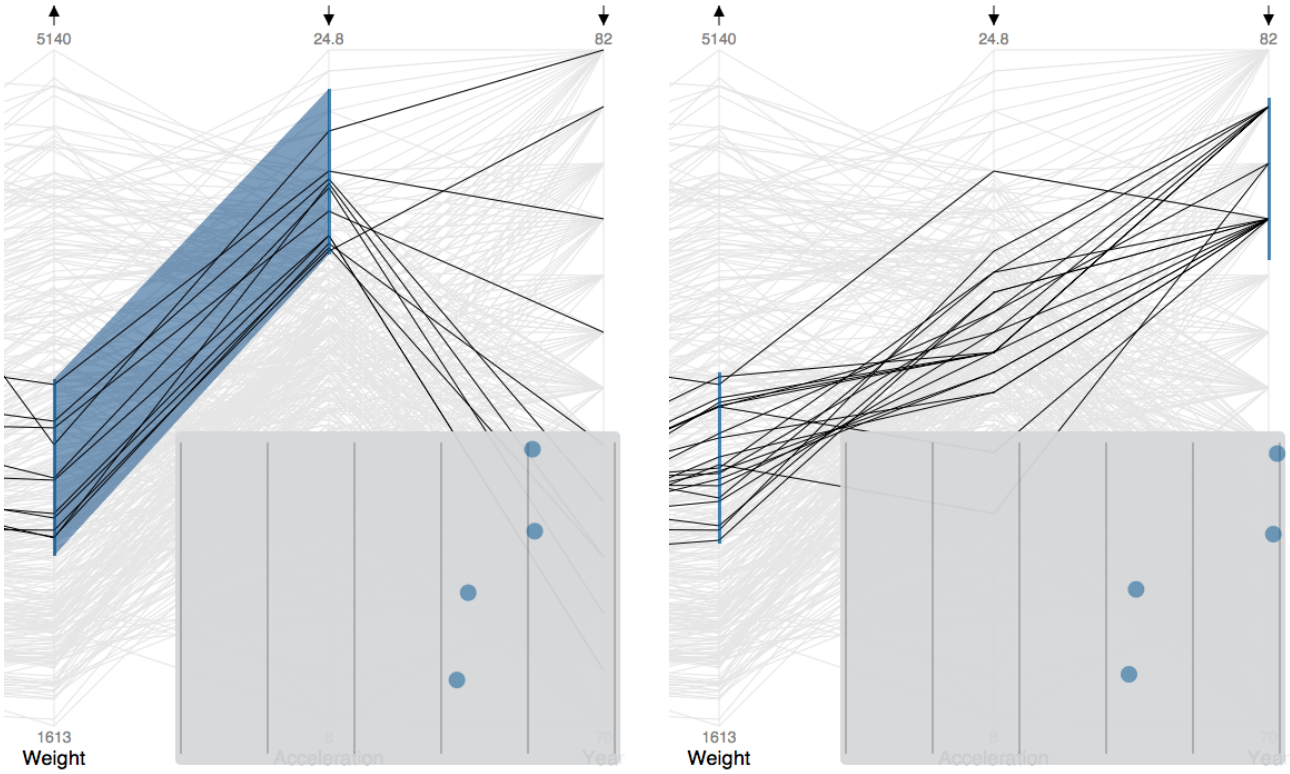
5.3 Three Fingers: Angular Brushing, Reorder Axes

Three fingers, with one on one axis and two on an adjacent one, select angular brushing (Figure 2b). This kind of brushing does not select based on a value range, but rather picks all data items whose line between the axes in question is at an angle that lies in the selected range of angles. The basis for this is a circle wedge that is drawn between the axes to indicate the range of angles covered by the brush.

In the original implementation of this feature,[4] the user has to pick three points to create an angular brush: the center point for the circle, the starting point of the arc, and its end point. This kind of brushing is much less predictable than single-axis brushing, and thus requires a lot more trial and error. It is easy to get frustrated by its tediousness and hit-and-miss nature, which reduces a useful tool to a theoretical feature that sees little actual use.

In our implementation, the user picks one point as the center of the circle on one axis and two points on the adjacent axis to select the start and end points of the arc. All three points (and thus the direction and size of the selection) can be moved simultaneously and independently. To show the user the selection, an arc is drawn between the axes, but without touching either of them. This is to not imply that the selection is anchored to positions on the axes, but happens between them.

A second interaction that is possible with three fingers is axis reordering. With all three fingers on the same axis, the user can move the current axis to another location by sliding her fingers across the trackpad; the other axes are rearranged accordingly. Three fingers are more difficult to position precisely than two, but this choice also makes accidental reordering much less likely. In typical use, the user only reorders axes occasionally, and spends much more time brushing.



(a) Brushing adjacent axes selects lines in the trapezoid.

(b) Four fingers independently brush two non-adjacent axes.

Figure 3: Multi-touch interaction for brushing in parallel coordinates with four fingers, on adjacent and non-adjacent axes. Insets show the positions of fingers on the trackpad.

5.4 Four Fingers: Brush on Two Axes

Four fingers are the logical extension of two fingers: brushing on two different axes at the same time, using two fingers each. The first brush still sticks to its axis, but the other one can be moved around freely. There are two distinct cases here: brushing adjacent axes and brushing non-adjacent axes.

When the axes are adjacent, the program draws a trapezoid between the two brushed areas (Figure 3a). This trapezoid encloses all brushed lines between the axes, and thus serves as a visualization of the brush itself. This is similar to the brushing in the time-series visualization tool TimeFinder,[13] where users can select ranges on different time steps that lines have to pass through to be selected. Our approach is much more interactive, but could be easily applied to this kind of data.

When axes are not next to one another, the two brushes are simply drawn on each axis separately (Figure 3b). A common problem in parallel coordinates is that correlations between non-adjacent axes are hard to see. Using simultaneous brushing on two axes, the user can quickly explore potential correlations (e.g., how many values go from low to high, how many from high to low) without having to rearrange the axes (which requires a reorientation).

Multiple brushes are possible in some programs, but have to be created sequentially and typically need to be deleted and recreated if they are to be changed. These programs typically allow the user to create more than two brushes, but we question the practical usefulness of more than two independent brushes at the same time – this might be useful for in-depth analysis, but it would be difficult for a user to keep track of more than two selections while exploring data. Our approach also easily degrades to the conventional single-mouse interaction if needed.

6. IMPLEMENTATION

Our prototype is implemented using Apple's Cocoa Objective-C libraries, utilizing the new `NSTouch` class supported in Mac OS X 10.6, *Snow Leopard*. The only information needed are the x and y coordinates of the touches, as well as an identifier that lets us track individual touches across touch events. We expect that kind of information to be available in most multi-touch frameworks, making our code easily portable. We have published our source code under an open source license for others to use[†].

After each change in the touches (touches being added, removed, or moving), the program first decides which axis each touch belongs to. It then performs different actions depending on the number of touches. In all but the single-finger case, if there was an active axis, it gets priority over a correct mapping of the fingers in their current location. The two- and four-finger cases directly call the brush functions that determine which points are to be highlighted, and then update the display accordingly. In the three-finger case, the program has to first decide whether all fingers are on the same axis, and if so, track their positions to the new location to then reorder the axes. If not, the angular brushing function is called to select data points.

7. DISCUSSION, LIMITATIONS

Multi-touch interaction on a comparatively small trackpad clearly has its limitations: only a small number of axes can be easily mapped and navigated by the user, and there are limits to the resolution of the touches as compared to the number of pixels. In practice, these do not limit the usefulness of the technique. While a large number of dimensions can be shown in parallel coordinates in principle, more than ten dimensions rarely make sense at the same time. The vertical resolution is also more than sufficient to pick ranges of two to three pixels on a 1280x800 pixel display. The intended use of this interaction, exploration, does not require pixel-level precision; the trackpad's resolution is also higher than the precision with which one can place a finger, which is the more natural limitation in this case.

Stability in the interaction is extremely important. The user has no direct way of knowing where her fingers are in relation to the mapped axes, and thus can easily move between them. This can cause unintended and distracting jumps in the interaction that are frustrating and take the user's attention away from the display. We therefore keep the first selected axis constant during an interaction, independently of the user's finger position. Angular brushing only makes sense on adjacent axes, so in this case we ignore the horizontal position of the user's fingers almost entirely.

When brushing two axes, the second axis responds to the user's finger position to make switching between axes easier. It is quite common to be focusing on one axis and wanting to find out which other axes have positive or negative correlations with it (e.g., looking at the weight of cars: When were lighter/heavier cars made? What is the relation to acceleration?). There is a trade-off here between stability and flexibility, and we believe that our choices made based on our own experience provide a good starting point.

While we have not formally evaluated the technique, feedback from our colleagues and our own experience with visualization tools indicates that the gained ease, speed, and immediacy of interaction is extremely helpful. Most visualization tools only provide rather basic interaction, and often require many mode switches for even simple tasks. Our multi-touch interaction does not cover all possible tasks, but it does significantly speed up the most common ones in exploring a dataset, and eliminates the required mode switches.

8. CONCLUSIONS AND FUTURE WORK

Multi-touch interaction provides a new way of exploring data visualizations, in particular parallel coordinates. The speed and ease with which the user can navigate the dimensions and potentially interesting patterns in the dataset facilitates exploration and analysis of data.

The work described in this paper is only a first step towards developing meaningful multi-touch interactions for visualization. The next steps include developing similar interaction techniques for different visualization techniques, as well as performing a series of user studies to better understand and calibrate the technique. We believe that multi-touch interaction not only provides a faster and easier way to interact with data, but also encourages users to spend more time exploring data in a visualization.

[†]`http://github.com/eagereyes/ParVisMT/`

REFERENCES

[1] Inselberg, A. and Dimsdale, B., "Parallel coordinates: A tool for visualizing multi-dimensional geometry," in [*IEEE Visualization*], 361–378, IEEE CS Press (1990).

[2] Inselberg, A., [*Parallel Coordinates: Visual Multidimensional Geometry and Its Applications*], Springer (2009).

[3] Siirtola, H., "Direct manipulation of parallel coordinates," in [*International Conference on Information Visualisation (IV2000)*], IEEE Computer Society Press (2000).

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

[5] Benko, H., Wilson, A. D., and Baudisch, P., "Precise selection techniques for multi-touch screens," in [*Conference on Human Factors in Computing Systems (CHI)*], 1263–1272, ACM (2006).

[6] Moscovich, T. and Hughes, J. F., "Indirect mappings of multi-touch input using one and two hands," in [*Conference on Human Factors in Computing Systems (CHI)*], 1275–1284, ACM Press (2008).

[7] Forlines, C., Esenther, A., Shen, C., Wigdor, D., and Ryall, K., "Multi-user, multi-display interaction with a single-user, single-display geospatial application," in [*Proceedings of the 19th annual ACM symposium on User interface software and technology (UIST)*], 273–276, ACM (2006).

[8] Schmidt, D., Block, F., and Gellersen, H., "A comparison of direct and indirect multi-touch input for large surfaces," in [*INTERACT*], 582–594, ACM Press (2009).

[9] North, C., Dwyer, T., Lee, B., Fisher, D., Isenberg, P., Inkpen, K., and Robertson, G., "Understanding multi-touch manipulation for surface computing," in [*Proceedings INTERACT*], 236–249, Springer (2009).

[10] Isenberg, P. and Fisher, D., "Collaborative brushing and linking for co-located visual analytics of document collections," *Computer Graphics Forum (Proceedings EuroVis)* **28**(3), 1031–1038 (2009).

[11] Han, J. Y., "multi-touch sensing through frustrated total internal reflection," in [*Proceedings User Interface Software and Technology*], 115–118, ACM Press (2005).

[12] Shneiderman, B., "Direct manipulation: A step beyond programming languages," *IEEE Transactions on Computers* **16**(8), 57–69 (1983).

[13] Hochheiser, H. and Shneiderman, B., "Visual specification of queries for finding patterns in time-series data," in [*Proceedings of Discovery Science*], 441–446, Springer (2001).