# Poster: Indirect Multi-Touch Interaction for Brushing in Parallel Coordinates

Robert Kosara*

**ABSTRACT**

Interaction in visualization is often complicated and tedious. Brushing data in a visualization such as parallel coordinates allows the user to select data points according to certain criteria; modifying a brush requires a lot of effort and mode switches.

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.

## 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.

Brushing in parallel coordinates [3] usually means selecting a range of values on one or more axes. There are also extensions of this idea, such as angular brushing [1], which lets the allow brushing 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 Pro and desktop computers with Apple's new *Magic Trackpad*. 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 INDIRECT MULTI-TOUCH INTERACTION

While multi-touch interaction has received a lot of attention lately, most of it is done by directly touching the display surface. Indirect (multi-)touch interaction is rare, and tends to focus on large surface [4] and recreating the interaction on a different surface [5].

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 switch from interaction to looking to interaction distracts from the exploration of the data.

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

## 3 MULTI-TOUCH FOR PARALLEL COORDINATES

The central idea of this work is the mapping of areas on the trackpad to axes on the display. 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.

### 3.1 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.

### 3.2 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 1a). 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 the brushing. The brush sticks to that axis until the user goes back to one or no fingers on the trackpad (this also persists through the three- and four-finger interactions below).

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 often means deleting the existing brush or at least replacing it with a new brush interaction that starts from scratch. Being able to easily change the position and size of the brush speeds up selection and requires less attention to the interaction.

### 3.3 Three Fingers: Angular Brushing, Reorder Axes

Three fingers, with one on one axis and two on an adjacent one, select angular brushing (Figure 1b). 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.

In the original implementation of this feature [1], 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. The sheer tediousness of the original implementation discourages users from using it.

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.

A second interaction that is possible with three fingers is axis reordering. With all three fingers on the same axis, the user can move

*UNC Charlotte, e-mail: rkosara@uncc.edu

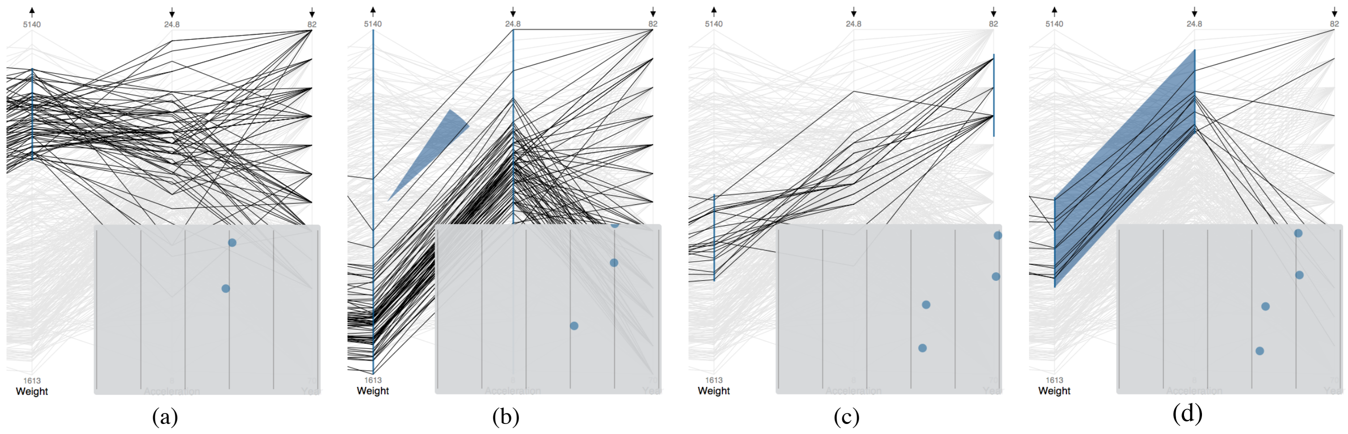

Figure 1: Multi-touch interaction for brushing in parallel coordinates: a) two fingers brushing on a single axis; b) three fingers performing angular brushing on adjacent axes; c) four fingers independently brushing two non-adjacent axes; d) four fingers brushing adjacent axes, only lines in the blue trapezoid are brushed. Insets show the positions of fingers on the trackpad.

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.

### 3.4 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 axes are not adjacent, the two brushes are simply drawn on each axis separately (Figure 1c). 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).

When the axes are adjacent, the program draws a trapezoid between the two brushed areas (Figure 1d). 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 [2], 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.

## 4 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.

## 5 Discussion, Limitations

Multi-touch interaction on a relatively 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 a few pixels on a 1280x800 pixel display. The intended use of this interaction, exploration, does not usually require pixel-level precision.

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 users attention away from the display. We therefore keep the first selected axis constant during an interaction, independently of the users 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.

While we have not formally studied 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 rudimentary 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, and eliminates the required mode switches.

## 6 Conclusions

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

The source code for our research prototype is available from `http://github.com/eagereyes/ParVisMT`

## References

[1] H. Hauser, F. Ledermann, and H. Doleisch. Angular brushing of extended parallel coordinates. In *Proceedings Information Visualization*, pages 127–130, 2002.

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

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

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

[5] D. Schmidt, F. Block, and H. Gellersen. A comparison of direct and indirect multi-touch input for large surfaces. In *INTERACT*, pages 582–594. ACM Press, 2009.