# Zliding: Fluid Zooming and Sliding for High Precision Parameter Manipulation

*Gonzalo Ramos, Ravin Balakrishnan*
Department of Computer Science
University of Toronto
bonzo, ravin@dgp.toronto.edu

**ABSTRACT**
High precision parameter manipulation tasks typically require adjustment of the scale of manipulation in addition to the parameter itself. This paper introduces the notion of Zoom Sliding, or *Zliding*, for fluid integrated manipulation of scale (zooming) via pressure input while parameter manipulation within that scale is achieved via x-y cursor movement (sliding). We also present the Zlider (Figure 1), a widget that instantiates the Zliding concept. We experimentally evaluate three different input techniques for use with the Zlider in conjunction with a stylus for x-y cursor positioning, in a high accuracy zoom and select task. Our results marginally favor the stylus with integrated isometric pressure sensing tip over bimanual techniques which separate zooming and sliding controls over the two hands. We discuss the implications of our results and present further designs that make use of Zliding.

**Categories and Subject Descriptors:** H.5.2 [User Interfaces]: Interaction styles; I.3.6 [Methodology and Techniques]: Interaction techniques.

**General Terms:** Design, Experimentation, Human Factors.

**Additional Keywords and Phrases:** input, pen-based interfaces, pressure widgets, multi-scale navigation.

## INTRODUCTION

The manipulation of a parameter is a fundamental task in most user interfaces. Although high precision parameter manipulation can be accurately achieved by simply entering an exact numeric value with an appropriate text input technique, from the user's point of view this exact method is not always the most appropriate or preferred. Interactions such as identifying and then picking a single pixel from a high resolution image, seeking a particular frame in a long video stream, or adjusting a continuous image color parameter are examples of parameter manipulation tasks where more interactive direct manipulation techniques can be preferable since the user may not be certain a-priori as to what value to enter. Furthermore, the immediate feedback that an interactive widget or technique can provide while the user adjusts the parameter is immensely valuable as it affords a more continuous style of interaction rather than the discrete style that results when specific values are entered explicitly. The challenge in designing interactive techniques for continuous high precision parameter manipulation is that the manipulation scale desired by the user when adjusting parameters may differ from one parameter to another, or even within the same parameter in different usage scenarios. Thus, interaction techniques for high precision parameter manipulation should support fluid adjustment of the scale within which the manipulation occurs, allowing users to make coarse scale manipulations for initial adjustments followed by finer scale manipulations for the final precise parameter specification.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
*UIST'05*, October 23–27, 2005, Seattle, Washington, USA.
Copyright 2005 ACM 1-59593-023-X/05/0010…$5.00.

In this paper, we propose and study a mechanism for use with pressure sensitive input devices, called Zoom Sliding, or *Zliding* for short, in which users use the pressure modality to fluidly and explicitly *zoom* or adjust the granularity of the parameter space, while *sliding* or dragging the input device to perform high precision parameter manipulation within that zoomed parameter space. We review the literature, discuss our design goals, propose and develop an interface widget for Zliding called the Zlider (Figure 1), and present a controlled experiment that examines how a Zlider can be used with three different input strategies for high precision parameter manipulation with concurrent control of adjustment granularity. We conclude by discussing the implications of the experiment's results on high precision parameter manipulation and propose design variations for alternate Zliding widgets.

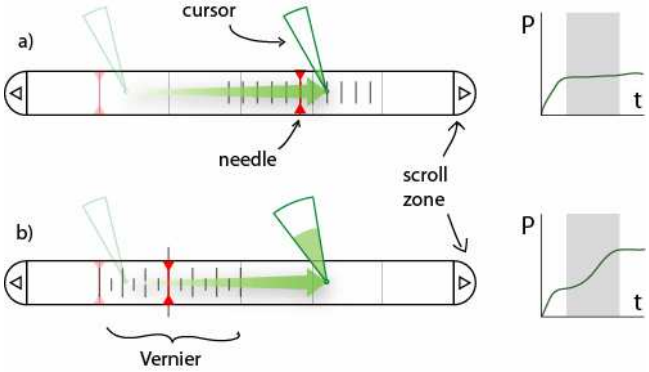


Figure 1: Zliding on the Zlider widget. a) A user manipulates a parameter at coarse granularity by sliding through the control while applying low pressure with the pressure transducer. b) The same x-y sliding action while pressing harder increases the granularity of the parameter space, allowing for more precise parameter manipulation when desired. The graphs on the right plot pressure over time, with the interval when the sliding occurs highlighted.


APPLE 1004

## RELATED WORK

Common strategies to facilitate spatial parameter selection tasks include reducing its Fitts' *Index of Difficulty* (ID) [14] by making targets larger, or by bringing them closer to the user's pointer. McGuffin et al. [25] show how increasing a target's size even at the final stages of a pointing task can be beneficial. Drag-and-Pop [7] reduces the distance that a user has to travel by bringing objects closer, using the pointer's trajectory. Another strategy to increase the precision of the user's interaction is to adjust the input device's control-display (CD) ratio. Semantic Pointing [9] improves the selection of objects by assigning them different CD ratios according to their importance. However, both these elements are fixed, which could be problematic if the user's assessment of what is important changes. All these approaches are aspects of the same solution: changing the scale of a target or the space that contains it. In some instances, zooming occurs in the visual domain, and in others, the motor domain [9].

There has been a consistent effort to develop controls and interactions tailored for precise parameter selection and manipulation tasks. The Alphaslider [2] is a compact selector that allows users to quickly pick a single item from a list of thousands, essentially by providing 3 sub-sliders with different levels of granularity. The FineSlider [30] extends the Alphaslider's idea and lets users adjust the rate at which the slider's selection changes, by using a rubber-band metaphor. The PVSlider [27] also uses a rubber-band metaphor to adjust the granularity with which users slide through a video stream. The issue of precise manipulation also applies to scenarios where the input mechanism can be imprecise by nature. Potter et al. [26] investigate how to increase the accuracy of a bare finger on a touch screen and show how their "take-off" approach outperforms traditional touching techniques. With "take-off", a target is defined not by the position a finger lands on, but by the position it is lifted at. This lets users adjust a cursor's position while their finger stays in contact with the touch screen. However, this approach does not allow for the gain of the finger's movement to be changed. Albinsson and Zhai's *Precision Handle* [3] lets a user's bare finger manipulate a graphic handle around a pivot in order to change the interaction's granularity and achieve pixel-level accuracy.

Guiard et al. [19] recognize that high precision selection tasks can be thought of as multi-scale navigation tasks. In addition, there is a significant body of work establishing a comprehensive theoretical framework for multi-scale tasks. Furnas' space-scale diagrams [16] give us the means to understand and analyze multi-scale interactions and interfaces. Guiard et al. [18, 19] have also shown that multi-scale pointing still obeys Fitts' Law. Building on this evidence, we believe that facilitating space scaling and manipulation operations should help users with high precision tasks. Some of the literature suggests that panning (i.e., equivalent to parameter manipulation) and zooming is an integrated task [23], and recommends it be driven by an integrated device. Igarashi and Hinckley [22] introduce speed-dependent automatic zooming, a technique that facilitates navigation tasks over large spaces, using a 2dof integrated device. This technique keeps the visual flow of the navigation constant, while scrolling at different speeds, thus improving users' performances over traditional scroll and pan and zoom methods [13]. However, other results indicate that there may be benefits in separating pan from zoom. It has been shown how bi-manual interaction techniques can be faster [12] than uni-manual ones, and can permit parallelism in multi-scale tasks [10, 21, 31].

A common theme present in all the above uni-manual designs and techniques is that both scale and parameter values are specified as a function of the cursor's x-y position. Further in many of these techniques, scale adjustments are determined by the system without giving users much say as to what scale values to apply and when. With bimanual techniques, the scale is often controlled by the non-dominant hand, while parameter manipulation within that scale is performed with the dominant hand. In contrast, our current work focuses on how users can control scale via a pressure transducer while simultaneously manipulating a parameter within that scale space using a spatial x-y cursor.

## MOTIVATION and GOALS

In our research, we are motivated by the steady technological progress in pen-enabled and touch-sensitive platforms, where high precision manipulation tasks are made even more challenging by very small or very large physical form factors and interfaces. The fixed granularity of standard GUI widgets like sliders may work reasonably well in a desktop computing environment, but may not scale to tiny PDAs or very large wall sized displays. On a tabletPC or PDA with small screen and input space, for example, the fixed relatively coarse granularity of some GUI widgets can hinder user's ability to make high precision adjustments. Example scenarios where users may need to fluidly adjust the scale of the parameter space in order to make precise parameter adjustments include:

- *Graphic applications*: Users may need to quickly select a precise pixel from a large bitmap that cannot be displayed at pixel-visible resolution on a small screen. Also, users may need to precisely adjust a value controlling a visual feature, such as the blending across several images.
- *Browsing on ZUIs*: Users may need to navigate through a map both at a very large and at a very fine scale.
- *Acquisition of small controls in the GUI*: Elements in an interface can present very small selection footprints, requiring a change in CD ratio to facilitate selection.
- *Analog-like controls*: These controls offer a granularity that depends on their physical size and the input device's CD ratio. Users interacting with such controls may need to do fine tuning in order to attain a precise value, such as a frequency in a radio tuner.

In designing a fluid interaction mechanism that facilitates high precision parameter manipulation, we have the following goals:

- *Integrated scale and parameter manipulation*. The interaction should support zooming of the parameter's scale space and concurrent high precision adjustment of the parameter within that space.
- *Infinite parameter scale adjustment.* It should be possible to fluidly adjust a parameter to an infinitely small or large value. The ability to attain virtually infinite precision or gain is a rarely explored objective that we believe to be worthy of attention.
- *Familiar interactions.* The new interaction should feel familiar, leveraging the typical user's vast experience with standard GUI widgets and interaction techniques.

## THE ZLIDER

The Zlider widget (Figure 1) consists of a rectangular working area that the user can scrub in order to adjust a parameter $v \in [low, high]$, where *low* and *high* are arbitrary limit values. There is no particular handle the user needs to grab to use the widget. To operate the Zlider the user taps and drags its pointer across the working area until the desired value is reached or effect is achieved. At all times a red needle indicates the position of the value being adjusted relative to the possible minimum and maximum values at the extremes of the widget. The Zlider's *scale* or granularity at time *i* dictates how the parameter *v* changes: $v_i = v_{i-1} + \Delta.range/(scale.length)$; where $v_0 = low$, $\Delta$ is the distance between the tapping point and the pointer's current dragging location, *range* = *high-low*, and *length* is the working area's length. The Zlider also displays a Vernier as suggested by Ayatsuka et al. [5]; however, the Vernier in our Zlider adapts its grid spacing depending on the widget's current scale factor.

### Pressure Cursor

Though not integral to the Zlider design, we use a pressure cursor (Figure 2) across our implementations, instead of the default cursor found in most GUIs. Our pressure cursor provides users with a real-time indicator of the pressure they are applying with the input transducer. The pressure cursor has a wedge-like shape that changes its aperture with the amount of pressure applied. The wedge's area fills as the pressure increases, until completely filled when the pressure reaches the maximum level the device can sense. The cursor's hot spot corresponds to the wedge's vertex.

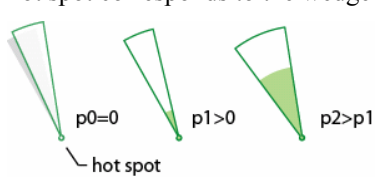


Figure 2: Pressure cursor. The wedge increases in size and fills up as pressure increases. (p2>p1>p0)

### Integrated Zoom & Slide Control

Our default interaction design uses a pressure-sensitive stylus as an integrated input device for fluid zooming of the parameter's scale space and sliding (i.e., manipulation) of the parameter's value within that scale space. The scale factor of the Zlider is adjusted by changes in pressure at the stylus' tip, and the stylus' x-y position enables sliding of the parameter's value. We use an exponential function of the form $scale=base^{f(p)}$ to calculate the scale factor, where $f(p)$ is a function of the stylus' reported pressure at a particular time.

Previous research in pressure-enabled widgets [28] highlights the difficulty users can experience in maintaining a constant level of pressure while dragging a stylus. We therefore utilize a combination of both signal processing and interactive techniques to minimize unwanted changes in the control's scale. Raw pressure data first passes through a low-pass filter. Then, it passes through a hysteresis process that stabilizes the signal further. Finally, a parabolic-sigmoid transfer function is used to account for users' performance when they apply force through an isometric input device like the stylus' tip pressure sensor. This transfer function has been used in similar scenarios [6] and is consistent with the effect we want to achieve. This effect is comprised of an initial "dead zone", slow response at low pressure levels (where users can vary pressure significantly without noticing), linear behavior in the mid ranges (where users have good control of pressure), and slow response at high levels of pressure (where the user's applied force can produce tremors, causing sudden pressure variations that are magnified by the exponential scale function). Figure 3 shows an example of how the different pressure-stabilizing stages affect the resulting scale factor.

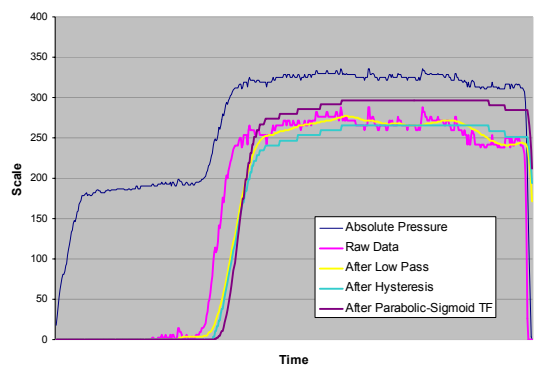


Figure 3: Effect of the stabilization techniques. The absolute pressure line represents the transducer's raw signal.

### Clutching the Zoom Level

The Zlider design has a clutching mechanism that enables users to completely stabilize pressure and hence lock the zoom level while sliding. Users clutch by sliding the cursor away from the Zlider's working rectangle (Figure 4.2). While clutched, users can still slide outside the working rectangle (Figure 4.3) but the widget maintains its scale at the last reported value regardless of pressure variations. Re-entering the working rectangle declutches (Figure 4.4).

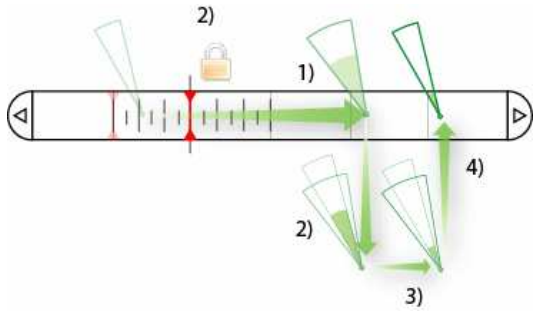


Figure 4: Clutching the zoom level

While the widget's scale factor is the same at the point in time when users clutch and declutch, it is possible that the pressure they applied at these moments is not. By design, we use this situation to let users increase the Zlider's scale factor arbitrarily in a relative manner. In other words, by clutching, users can not only stabilize scale variations, but also achieve as much precision as needed. To go beyond the scale value attainable when the pressure at the stylus' tip reaches it maximum value, users can in one continuous gesture: a) increase pressure and hence scale factor; b) clutch; c) decrease applied pressure; d) declutch; and e) increase pressure and hence scale beyond the value at step (a). This process can be repeated in order to attain higher precision levels if the user so desires. Conversely, an inverse series of steps allows users to decrease the scale factor from a high to a low level.

There are three ways in which the user is notified that they are clutching or declutching: a) a very brief auditory feedback, b) an icon that follows the Zlider's needle (Figure 4.2), and c) a change in the physical appearance of the pressure cursor (Figure 4.2-3). Pilot studies revealed that while visual feedback is important, auditory feedback was beneficial to users who were not visually focusing on the Zlider control.

**The Selection Mechanism**

The Zlider design uses the release of the stylus from the interaction surface as an indication of selection. This is consistent with the behavior of regular slider controls, and previous research [28] also supports lifting the stylus as a selection technique for pressure-aware widgets. However, some issues remain that deserve our attention. First, we need to determine the Zlider's behavior when the stylus is lifted from the interaction surface (i.e. the applied pressure becomes zero). Even though one possible design decision is to make the scale=1, this is not always desirable. Pilot studies revealed that users might lift the stylus because they wanted to re-invoke the Zlider from a different point when they found themselves sliding very close to either extreme of the working rectangle. Users indicated that resetting the scale to 1 was annoying, since it forced them to reacquire the scale value. The same situation was found when users missed the target parameter value by a small amount. In this case, they explicitly voiced the need to perform, as one user called it, "quick micro-adjustments". Based on this feedback, we modified the Zlider's behavior so that it maintains its last reported scale value as long as the stylus is within sensing proximity and the widget's working area. This sensing or tracking capability can be found in most modern digitizing tablets as well as in other display technologies (such as the SmartBoard), and has started to be used as a design element in a number of novel user-interface widgets [8, 15]. It is possible to use time-based techniques to simulate to some degree this behavior in devices that lack proximity sensing. However, a full discussion is beyond the scope of this paper. Figure 5 shows the state-transition diagram of the Zlider's behavior.

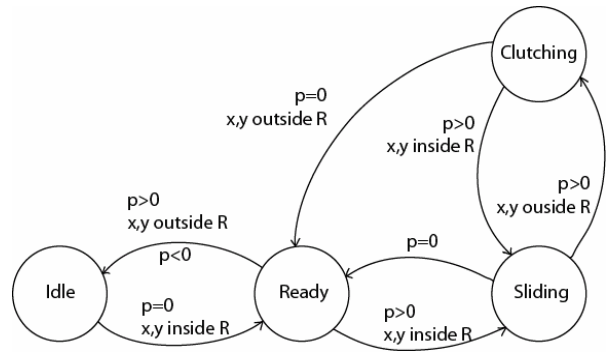


Figure 5: Zlider's state-transition diagram. *R* is the working rectangle; x,y the cursor's position. *p<0*: stylus out of range; *p=0*: stylus is being tracked; *p>0:* stylus is touching the tablet. Zlider's scale is reset to 1 at the idle state.

The second issue we need to consider is estimating what the Zlider's last reported scale (pressure) value should be at the time the stylus is lifted from the interaction surface. We need to identify as accurately as we can the exact moment when users start their lifting action. This is important as we do not want the Zlider to accidentally change its scale factor. In our case, looking back a fixed number of samples, as was proposed by Buxton [11], is not sufficient because the number of samples we need to trace back depends on how fast users lift the stylus. Observations in our pilot studies also revealed that users generally pause for a few milliseconds before lifting the stylus, thus defining a very small pressure valley. Furthermore, pressure values from that point onwards follow a monotonically decreasing trend. With the above information we estimate the scale factor at the time the user starts to lift the stylus. Our algorithm looks backwards in the device's buffer until the small valley is found, or the curve stops its decreasing trend. Since the Zlider control's scale responds in real-time to variations, it is possible that there is a mismatch between the estimated scale value and the scale at the point the stylus is lifted. Sudden changes in the Zlider's scale factor would result in an undesired disorienting effect on the user. In order to mitigate this effect, the Zlider's scale smoothly changes to the estimated last reported scale value. The same type of smooth transitions is consistently used in the Zlider widget when changes, otherwise too abrupt, need to occur.

These two design features provide functionality similar to clutching, wherein users have another way to achieve an arbitrarily high level of precision. In this case, to increase the scale once no more pressure can be applied, a user can: a) increase pressure and hence scale; b) quickly lift the stylus from the interaction surface, staying within tracking distance; c) touch the working area again; and d) increase pressure until the desired magnification is achieved. Unlike clutching, this tracking interaction does not allow users to un-zoom in a controlled fashion. Nonetheless, we observed that both the tracking and clutching mechanisms served different users' interaction styles when adjusting the zoom.

### Scrolling

The Zlider is controlled by relative displacements in its working area. However, pilot studies showed that some users wished the familiarity of continuous scrolling found in ordinary scroll and slide controls. Our design easily incorporates continuous scroll zones at the extremes of its working area (Figure 1). If, while sliding, the cursor reaches a scroll zone the Zlider enters a scrolling mode. Sliding has no effect in this mode and the parameter it controls changes at a constant rate proportional to the current scale. By adjusting pressure, scale can be changed while in scroll mode thus affecting the scrolling speed.

## ALTERNATIVES FOR DECOUPLED ZOOM CONTROL

The Zlider was designed to be operated by an integrated pressure and position sensing input device, such as a pressure-enabled stylus. However, our design can easily support other ways to adjust the Zlider's scale factor. In particular, we can use input originating from the user's non-dominant hand. Decoupling the scale control from the dominant hand has the potential to eliminate undesired interference between zooming and panning that may occur while using the stylus as the only input device. At the same time, this decoupled way of controlling scale has the potential to still allow users to perform zooming and sliding concurrently [10]. In this section we explore two instances of decoupled design strategies for adjusting the Zlider's scale: a force sensing button, and two discrete keys.

### Force Button

A force button is an isometric input device that can have a minimal footprint. This makes it an attractive design choice that can be incorporated in many form factors such as hand-held devices, tablets, and even in traditional input devices such as mice or keyboards. In addition, previous research [20] shows the potential advantages of embedding force sensors on hand-held devices. For our exploration of this style of input we used a phidget [17] force sensor. The signal reported by this sensor is very similar to the one given by the stylus' tip and we use it in the same way. Because of this similarity, many of the issues regarding signal stabilization that we discussed in the previous sections apply to this input device. However, since the force button is decoupled from the stylus, it is easier to determine what the scale factor is at the time users lift the stylus. Nonetheless, we found that both the signal stabilization techniques and clutching mode already discussed were effective at mitigating signal instabilities while users slide. Clutching and tracking can be used with this input mode to achieve arbitrarily high precision levels.

### Discrete Keys

The second decoupled method of controlling the Zlider's scale uses two discrete keys: one for increasing the scale and another for decreasing it. This input mechanism is easy to implement in a variety of form factors and sizes and it can be seen as the lowest common denominator method for changing the scale in many scenarios. We implement this input mode using the *Shift* and *Crtl* keys found in most computer keyboards. Users tap on *Shift* and *Ctrl* in order to respectively increase or decrease the scale factor by a constant increment. Also, users can tap and hold on either key in order to zoom or un-zoom at a continuous rate. The signal from this input is stable, making it easy for users to slide at constant scales. Consequently, we neither need to filter the input, nor use the parabolic-sigmoid transfer function. Also, finding the scale value when the stylus is lifted becomes trivial. Though clutching and tracking are still available, users can use the keys alone to reach arbitrarily high precision levels. However, this discrete input has a drawback in the amount of time a user requires to reach a determined scale factor. This time depends both on the mechanical properties of a key that needs to be pressed and released, and the rate at which scale is adjusted when a key is held. This rate needs to be carefully considered. A rate that is too fast will make the interaction quicker, yet difficult to control (i.e. users will overshoot the desired scale). Conversely, a slow rate will make the interaction more controllable, yet unacceptably sluggish. Equally important is the choice of the step the scale should change for each keypress. For our experiments we updated the scale every 30ms after a key was held for 400ms.

## EXPERIMENT

Our experiment investigates how three different scale adjusting strategies: *Stylus*, *Force Button*, and *Keys* affect people's interactions and performance in a high precision selection task that uses the Zlider. We are particularly interested in investigating how these strategies compare to one another. In particular, we believe that the simplicity of the *Keys* and *Force Button* techniques could outperform the *Stylus* technique where the combination of linear x-y movement and pressure control with the stylus tip might interfere with one another. On the other hand, the integrated nature of the *Stylus* technique has the advantage in that users will likely conceive of the zoom and sliding task as a conceptual whole, rather than two separate subtasks as with the *Keys and Force Button* techniques where zooming and sliding control are separated across the two hands. Because each technique has its own idiosyncrasies, the experiment's results can help both designers and users to choose the best solution for a given situation. In addition, the experiment will provide us with valuable user feedback regarding the Zlider control and the overall experience of Zliding.

### Apparatus

We used a Wacom Cintiq 18SX interactive LCD graphics display tablet with a wireless stylus that has a pressure-sensitive isometric tip. The stylus reports 1024 levels of pressure, and has a binary button on its barrel. The stylus does not provide any distinguishable haptic feedback in relation to the pressure applied. The tablet's active area was mapped onto the display's visual area in an absolute one-to-one manner. To implement the *Force Button* condition we use a phidget [17] interface board that read data from a force sensor. Users applied force on the sensor through a thin layer of hard rubber protecting them from its uncomfortable original profile. Although this force sensor reports up to 1000 levels of force, we only use the first 2/3 of them, as in our pilot studies users showed discomfort when reaching values above 2/3 of the way. The *Keys* condition was implemented using the *Shift* and *Ctrl* key on a regular PC keyboard. The experimental software ran on a 1.4GHz P4 PC with Windows XP Professional.

### Participants

Four female and eight male volunteers, 18-44 years old, participated in the experiment. Ten were right-handed. All had little or no prior experience with tablets like the one used in the experiment. No compensation was provided.

### Task and Stimuli

A serial target acquisition and selection task was used. The stylus was used to control the sliding behavior of a Zlider widget with its scrolling zones disabled and its clutching and hover mechanisms enabled. The experimental trials simulate a pan and zoom task on a reduced interaction footprint, like the ones found in hand-held computers or dialog windows. In each trial the user controls the Zlider in order to locate and select a target in a workspace area 1500 pixels long, shown through a viewport 256 pixels long (Figure 6). The target to be selected is represented as a green rectangle and can have three possible widths: *1/10*, *1/1,000* and *1/100,000* of the workspace's length. In turn, the target can be located at a *near, mid* or *far* distance from the top of the workspace. Distance is chosen according to the target's width so that *distance=n.width*, where *n* is an integer, *width* is the target's width, and *distance* belongs to either the intervals *[150, 450)*, *[600, 900)* or *[1050, 1350)*. Besides the target, the workspace contains a horizontal grid that increases in density in the vicinity of the target, helping users locate it. As the user scrubs across the Zlider the workspace scrolls accordingly under the viewport in the same way a document scrolls in a text editor. During the trials users can adjust the interaction scale through one of three methods: *Stylus*, *Force Button* and *Keys*. Changes in the scale are reflected by magnification changes on the working area and on the stylus' C:D ratio. This scaling operation makes accessible targets that otherwise would be too small to select. Users are instructed to scroll through until the target is inside the viewport, visible, and covering the selection line (Figure 6). When this happens the target changes its color from green to red, and users finish the selection by lifting the stylus from the interaction surface. The workspace has a textured background, which helps users to be aware of the current scale factor they are at, thus alleviating desert fog [24] effects in the scale space.

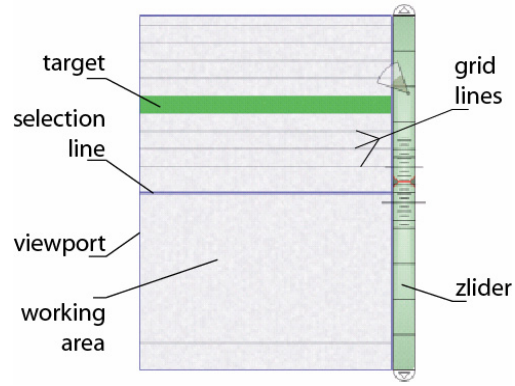


Figure 6: Elements in the experimental setup.

### Procedure and Design

A within-participants full factorial design with repeated measures was used. The independent variables were *Technique* (*Stylus*, *Force Button* and *Keys*), *Width* (*large, small, smallest)*, and *Distance* (*near, mid, far*). The order in which techniques were presented to users was included as a between-subjects factor. The dependent variables were *Selection Time* – defined as the time from the moment the stylus touches the tablet's surface until the moment the user selects the target; and *Crossings* – defined as the number of times the selection line enters and leaves the target per trial (e.g., this value is equal to 1 when a participant does not overshoot the target). *Crossings* gives us information about the degree of control shown by participants during a trial, as well as hints about their strategy during trials. For each experimental trial, we collected all the stylus, force button, and key data events. Also, since each trial can only be completed successfully, we end with a set of error-free selections. Participants were randomly assigned to 6 groups of 2 participants each. In each group, participants were exposed to all 3 *Technique* conditions, whose order of appearance was counterbalanced across groups to minimize ordering effects. For each *Technique*, participants were asked to complete 4 blocks each. Each block consisted of 9 selection trials (3 *Distances x 3 Widths*), repeated 5 times. Presentation of trials within a block was randomized. In summary, the experiment consisted of:

12 participants x
3 technique conditions x
4 blocks x
3 width conditions x
3 distance conditions x
5 repetitions
= 6480 target selection trials.

Prior to performing the trials for each *Technique*, the experimenter explained to the participant how the Zlider worked with a particular technique. Then participants did a warm-up block of 45 trials to practice with the corresponding technique. Participants were instructed to

perform the upcoming tasks as quickly and accurately as possible. While participants could take breaks between blocks, we enforced a 5 minutes break between techniques. A short questionnaire was administered at the end of the experiment to gather the participants' opinions.

## RESULTS

The experiment took an average of 1.25 hours per participant. A trial was considered an outlier when *Time* was beyond 2 standard deviations from the mean per participant. A total of 245 outliers (3.7%) were removed from our analysis. There were no main effects or interactions for the *Order* condition on either *Selection Time* or *Crossings*. While our data logs did record instances of clutching and hover, a detailed analysis at this level of interaction granularity is beyond the scope of this paper.

### Selection Time

As might be expected from Fitts' law, analysis of variance revealed a significant main effect on *Selection Time* for *Width* ($F_{2,14}$ = 392.8, $p$ < .0001), and *Distance* ($F_{2,14}$ = 13.23, p < .001). However, there were no significant main effect on *Selection Time* for *Technique* ($F_{2,14}$ = 0.31, $p$ = 0.738), or *Technique*Width* ($F_{4,28}$ = 1.53, $p$ = 0.22), and *Technique*Distance* ($F_{4,28}$ = 0.693, $p$ = 0.6) interactions. Also, post-hoc pairwise comparisons did not show any main effects between *Techniques* for all levels of the *Width* condition. This is an interesting finding because we did not expect users to perform statistically similarly with such distinct techniques. Figure 7-8 illustrate these results.

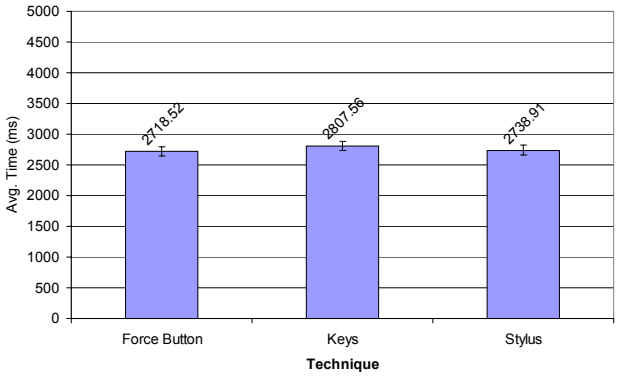


Figure 7: Average selection time per technique.

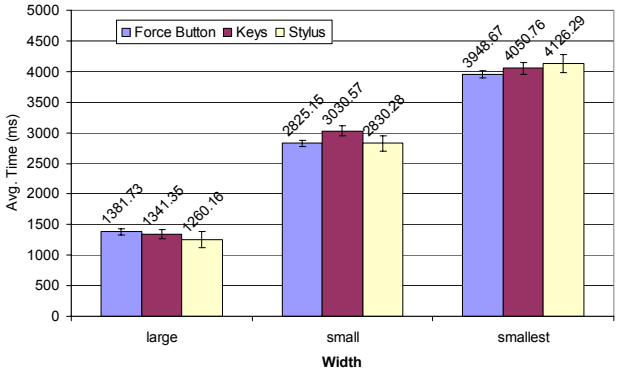


Figure 8: Average selection time per technique*width.

An analysis of *Selection Time* across experimental blocks (Figure 9) shows participants improving marginally as the experiment progressed for both the *Force Button* and *Stylus* conditions. For the *smallest* condition, participants' performance degraded and then recovered when using *Keys*, suggesting it may have taken longer for users using this technique to find a good strategy to complete a trial. Variations on the last experimental block suggest that fatigue effects may be present.

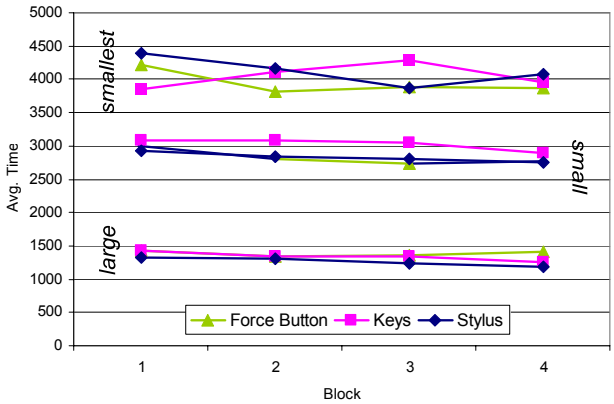


Figure 9: Average selection time per block*technique.

### Crossings

We found that participants crossed a target more than once for a number of reasons: a) they were sliding too fast and the target passed under the selection line without them noticing; b) they tried to acquire the target when it was visible, yet unreachable because the CD ratio was not high enough; c) fluctuations in their control of the scale caused the target to move; and d) a combination of all of the above. Our analysis shows a significant main effect for *Width* ($F_{2,4}$ = 357, $p$ < .0001) on *Crossings*. Pairwise comparisons indicate that *Crossings* at *mid* distances were significantly higher than crossings at *near* distances ($p$ < .03). There was no main effect for *Distance* ($F_{2,4}$ = 5.46, $p$ = .07). However, we observe for the *large* condition that crossings increase as distance decreases. Post-hoc comparisons indicate that *near* targets are crossed more often than *mid* ones ($p$ < .03). Figure 10 illustrates these effects.

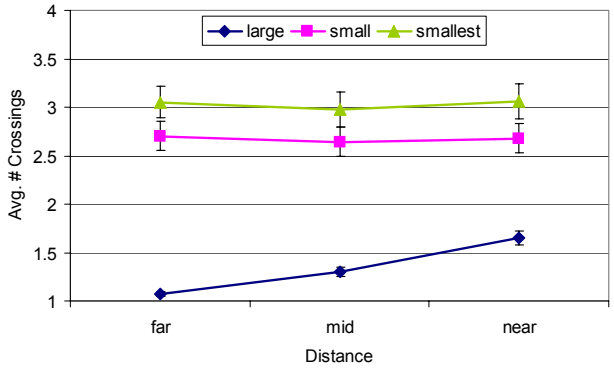


Figure 10: Average crossings per distance*width.

There was no main effect for *Technique* ($F_{2,4}$ = 2.24, $p$ = .22) on *Crossings*. However, analysis of variance shows a *Technique*Width* interaction ($F_{4,8}$ = 4.49, $p$ < .04) a closer inspection of the means shows *Stylus* resulting in fewer crossings than the other techniques for the *small* and *smallest* conditions. Figure 11 illustrates these effects.

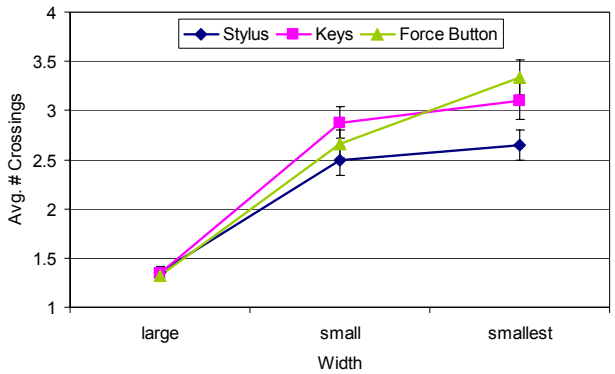


Figure 11: Average crossings per width*technique.

Figure 12 illustrates the number of crossings as the experiment progressed for each of the techniques. While participants do not seem to do better or worse with the *Stylus*, there is some improvement in participants' control for the *Force Button* with practice.

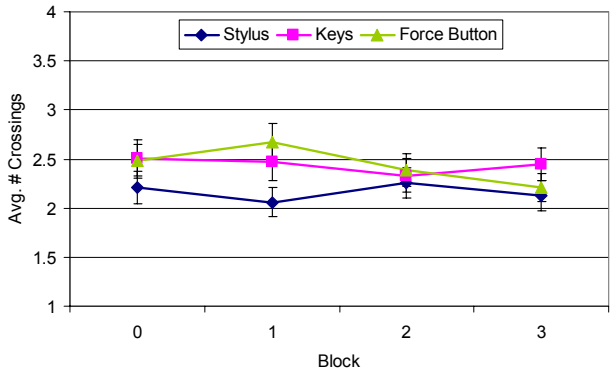


Figure 12: Average crossings per block*technique.

### Qualitative Results: User Preferences

At the end of the experiment, we asked participants to rank each of the techniques presented to them. Their responses revealed mixed opinions. While some participants preferred and showed more skill with the *Stylus*, others preferred the *Force Button* or the *Keys*. This was interesting, as the experimenter heard how much a participant "loved" a technique not long after another participant expressed her dislike for it.

Six participants ranked *Keys* as their 1st preferable method to use and ten people as their 1st to 2nd choice. Users gave several reasons for their ranking, e.g., "it was easy to learn", "it was simple to use", "it was predictable", "I could keep the scale stable". However, more than half the people in that group expressed that for many scenarios they would probably like to use the stylus alone, because it requires "only one hand" and "does not need a keyboard". It was not surprising to hear from some participants that "using the keys was slow", although the quantitative data does not support this claim. Only two participants ranked *Stylus* as their 1st choice and eight as their 1st to 2nd choice. Even though "it took longer to get used to" people expressed that once they "got it" the selecting task had a "cool fluid feeling" to it. While people in this group commented that it felt "quite natural" to zoom, they also expressed that it was challenging not to affect the pressure they are applying when sliding over long distances. This was a nuisance, if users did not want to alter the Zlider's scale while browsing for the target. One participant expressed that using the stylus alone was "incredibly fast when the target area was on sight". Four participants ranked the *Force Button* as their 1st choice, and six did so as their 1st to 2nd one. There was a mixed set of responses for this condition. While some people exhibited very good control, others did not. As was observed with the *Keys* condition, people in this group liked the fact that zooming and sliding were decoupled. However, people who did not like this condition complained about difficulties coordinating both zooming and sliding with separate hands. For example, we observed how users inadvertently accompanied the selection lifting action with a quick release of the *Force Button* as well.

## DISCUSSION

We had the opportunity to both assess a widget of our design, and to observe people using it to fluidly and successfully perform selection and micro-adjustment tasks in sizes that ranged from the large to the almost infinitesimal. The fact that we found no significant differences in terms of average *Selection Time* for the three scale adjusting *Technique* conditions we studied is both unexpected and remarkable. This result shows that the Zlider's design can be used in different scenarios and hardware configurations without any performance degradation.

Nevertheless, our results prompt us to consider metrics other than *Selection Time* in order to identify if a *Technique* is preferable. Our analysis of the number of *Crossings* per *Technique* favors the *Stylus* condition, which results in fewer crossings for small targets. This conclusion is reinforced by the participants' qualitative feedback, which not only helps us identify what works well with the *Stylus*, but also what can be improved. We believe that a critical area of Zliding requiring improvement is supporting users' ability to zoom only when they want to. Our Zlider design supports this feature with its clutching mechanism and by making use of the tracking capability of its input device. However it may be that Zliding needs to occur without an explicit physical area, or widget that can act as a clutching delimiter. Examples of these cases are a widget with no area, such as a crossing widget [4], or panning and zooming on a 2D map. It is then necessary to think of alternate strategies to achieve this design goal. Since most of the undesired scale changes were observed while the user was

dragging the stylus, one solution is to alter the rate at which the interaction's scale is allowed to change, based on the speed at which the stylus moves. This solution can include extreme cases such as disabling scale changes when the pointer moves above a certain threshold speed, or allowing scale changes only when the stylus is not moving in x-y space. Our current implementation is but a particular case of this general strategy. In addition, this solution provides users with an interaction style that models tasks that are purely serial (e.g., pan then zoom), purely coordinated (e.g., pan while zoom), or in-between.

### Other Designs: The Zliding Wheel

The Zliding Wheel (Figure 13) operates by the same principle as a knob control with the exception that one can control the granularity of the wheel's increments. This control provides functionality similar to the Zlider's, but with a potentially smaller footprint and no boundaries on the parameter it controls. With the Zliding wheel, in addition to using the curvature of the arc being drawn to regulate the granularity, users can also adjust it through a degree of freedom other than a cursor's position, such as the pressure applied with a stylus input device. We consider two main variations of the Zliding wheel: a fixed version (Figure 13-a), and a floating one (Figure 13-b).

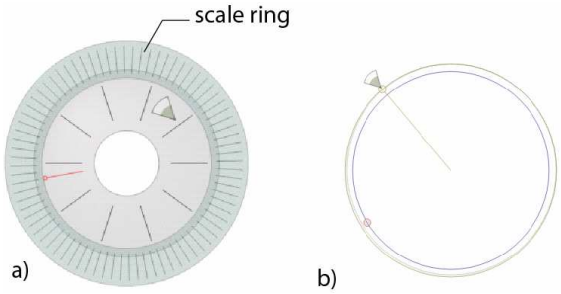


Figure 13: Zliding Wheels.

The fixed version (Figure 13-a) consists of a circular disk that users can rotate by scrubbing on its surface. A red needle inside the disk indicates the absolute rotation the disk is subjected to. The fixed Zliding wheel is very similar in its behavior to the Zlider widget: users can modify the wheel's granularity while they are rotating it, as well as access a clutching zone when they drag the pointer outside the disk's area. A scale ring is displayed above the wheel when its scale factor > 1, and provides a "gear-like feedback", which helps users understand the differences between the wheel's and the pointer's absolute motion.

The floating or "drifter" wheel (Figure 13-b) follows a pointer's circular motion, which is not necessarily centered on a fixed point, and works under the same principles of the control described in [29]. This drifting is usually the result of the user not focusing visually on the control but instead paying attention to the changes the control causes. Because of this, the floating wheel has a lightweight visual design that consists of two concentric rings: an internal one that provides the absolute rotation the wheel is subjected to; and an external one that keeps track of the pointer's current motion. The floating wheel provides minimal feedback about its granularity by altering the thickness of its outer ring. As it stands, this design cannot incorporate a clutching zone, and because of this, it would be appropriate to use the pointer's speed to determine when scale adjustments should be permitted.

It is worth mentioning that unlike other similar wheel controls, the Zliding wheel has the advantage of providing a way to adjust its granularity even in scenarios where it is not possible to move beyond the control's boundaries (e.g., a notebook's touchpad, or an iPod's scroll wheel).

## CONCLUSIONS AND FUTURE WORK

We believe that high precision parameter manipulation tasks can be greatly facilitated by allowing users to fluidly zoom and slide, or zlide, as they interact. Our results show that even though different scale adjusting strategies seem comparable in terms of elapsed time, there are advantages in the use of an integrated input device, such as the pressure-enabled stylus available in Tablet PCs. This is particularly useful in scenarios where a keyboard is not available or accessible. Users commented on how performing a selection using only one hand was appealing to them and "felt right". We also found that there is room for improvement in our interaction design. In particular, it is possible to develop more sophisticated filtering techniques in order to obtain a stable input signal from the stylus' pressure-sensitive tip. Nevertheless, we found it remarkable that, in spite of our simple signal filtering scheme, the stylus emerged as a reasonable alternative when compared with the other simpler decoupled input strategies. This result has implications that apply not only to pen-based systems, but also to a diverse set of devices and form factors capable of touch-based input.

### Other Technologies, Other Directions

There are other technologies that have the potential to be used to zoom and slide. For example, capacitive touchpads are becoming widespread as a means of input for notebook computers, portable music players, and handheld devices. Even tough capacitive touchpads have been traditionally used to sense a finger's position, some are capable of also sensing the amount of pressure that is applied to them. In other types of touch-sensitive surfaces, it is possible to estimate the contact area of a touching finger, thus estimating the applied pressure.

It is possible to leverage devices with decoupled continuous degrees of freedom such as Microsoft's Digital Media Pro keyboard or Wacom's Cintiq 21UX tablet, to facilitate the acquisition of very small targets in current GUIs. For example, in a fashion similar to the map application presented in [21], a user can apply scaling operations on a window's manager using a pointer's current location as a center of magnification. In this way, users can browse the GUI until a desired scale or CD ratio is reached. In a similar fashion, the "take-off" technique [26] can be greatly enhanced by Zliding. We imagine such interactions

becoming commonplace in new environments that use resolution-independent graphics as their rendering engine.

Future work includes additional interactions and widget designs that can take advantage of Zliding to facilitate high precision manipulation tasks. In particular, we would like to see how crossing widgets [1, 4] can incorporate Zliding techniques. Finally, we are interested in studying, in the context of Zliding tasks, the degree of coordination people exhibit when using both coupled and decoupled input strategies for a variety of form factors.


## ACKNOWLEDGMENTS

We thank all our experiment participants, and members of the DGP Lab (www.dgp.toronto.edu).


## REFERENCES


1. Accot, J., & Zhai, S. (2002). More Than Dotting the I's - Foundations for Crossing-Based Interfaces. *CHI*, p. 73-80.
2. Ahlberg, C., & Shneiderman, B. (1994). The Alphaslider: A Compact and Rapid Selector. *CHI*, p. 365-371.
3. Albinsson, P.-A., & Zhai, S. (2003). High Precision Touch Screen Interaction. *CHI*, p. 105-112.
4. Apitz, G., & Guimbretiere, F. (2004). Crossy: A Crossing-Based Drawing Application. *UIST*, p. 3-12.
5. Ayatsuka, Y., Rekimoto, J., & Matsuoka, S. (1998). Popup Vernier: A Tool for Sub-Pixel-Pitch Dragging with Smooth Mode Transition. *UIST*, p. 39-48.
6. Barrett, R., Olyha, J., Robert S., & Rutledge, J. (1996). Graphical User Interface Cursor Positioning Device Having a Negative Inertia Transfer Function. Patent # 5,570,111, IBM Corp.
7. Baudisch, P., Cutrell, E., Robbins, D., Czerwinski, M., Tandler, P., Bederson, B., & Zierlinger, A. (2003). Drag-and-Pop and Drag-and-Pick: Techniques for Accessing Remote Screen Content on Touch- and Pen-Operated Systems. *Interact*, p. 57-64.
8. Bezerianos, A., & Balakrishnan, R. (in press - 2005). The Vacuum: Facilitating the Manipulation of Distant Objects. *CHI*.
9. Blanch, R., Guiard, Y., & Beaudouin-Lafon, M. (2004). Semantic Pointing: Improving Target Acquisition with Control-Display Ratio Adaptation. *CHI*, p. 519-526.
10. Bourgeois, F., Guiard, Y., & Beaudouin-Lafon, M. (2001). Pan-Zoom Coordination in Multi-Scale Pointing. *CHI Extended Abstracts*, p. 157-158.
11. Buxton, W., Hill, R., & Rowley, P. (1985). Issues and Techniques in Touch Sensitive Tablet Input. *SIGGRAPH*, p. 215-224.
12. Buxton, W., & Myers, B. (1986). A Study in Two-Handed Input. *CHI*, p. 321-326.
13. Cockburn, A., & Savage, J. (2003). Comparing Speed-Dependent Automatic Zooming with Traditional Scroll, Pan and Zoom Methods. *British HCI*, p. 87-102.
14. Fitts, P. (1954). The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement. *Journal of Exp. Psychology*, *47*, p. 381-391.
15. Fitzmaurice, G., Khan, A., Pieké, R., Buxton, B., & Kurtenbach, G. (2003). Tracking Menus. *UIST*, p. 71-79.
16. Furnas, G., & Bederson, B. (1995). Space-Scale Diagrams: Understanding Multiscale Interfaces. *CHI*, p. 234-241.
17. Greenberg, S., & Fitchett, C. (2001). Phidgets: Easy Development of Physical Interfaces through Physical Widgets. *UIST*, p. 209-218.
18. Guiard, Y., Beaudouin-Lafon, M., Bastin, J., Pasveer, D., & Zhai, S. (2004). View Size and Pointing Difficulty in Multi-Scale Navigation. *AVI*, p. 117-124.
19. Guiard, Y., Beaudouin-Lafon, M., & Mottet, D. (1999). Navigation as Multiscale Pointing: Extending Fitts' Model to Very High Precision Tasks. *CHI*, p. 450-457.
20. Harrison, B., Fishkin, K., Gujar, A., Mochon, C., & Want, R. (1998). Squeeze Me, Hold Me, Tilt Me! An Exploration of Manipulative User Interfaces. *CHI*, p. 17-24.
21. Hinckley, K., Czerwinski, M., & Sinclair, M. (1998). Interaction and Modeling Techniques for Desktop Two-Handed Input. *UIST*, p. 49-58.
22. Igarashi, T., & Hinckley, K. (2000). Speed-Dependent Automatic Zooming for Browsing Large Documents. *UIST*, p. 139-148.
23. Jacob, R., Sibert, L., McFarlane, D., & Mullen, M. (1994). Integrality and Separability of Input Devices. *TOCHI*, *1*(1), p. 3-26.
24. Jul, S. & Furnas, G. (1998). Critical Zones in Desert Fog: Aids to Multiscale Navigation. UIST, p. 97-106.
25. McGuffin, M., & Balakrishnan, R. (2002). Acquisition of Expanding Targets. *CHI*, p. 57-64.
26. Potter, R., Weldon, L., & Shneiderman, B. (1998). Improving the Accuracy of Touch Screens: An Experimental Evaluation of Three Strategies. *CHI*, p. 27-32.
27. Ramos, G., & Balakrishnan, R. (2003). Fluid Interaction Techniques for the Control and Annotation of Digital Video. *UIST*, p. 105-114.
28. Ramos, G., Boulos, M., & Balakrishnan, R. (2004). Pressure Widgets. *CHI*, p. 487-494.
29. Tomer, M., & John, F. (2004). Navigating Documents with the Virtual Scroll Ring. *UIST*, p. 57-60.
30. Toshiyuki, M., Kouichi, K., & Borden, G. (1995). Elastic Graphical Interfaces to Precise Data Manipulation. *CHI*, p. 143-144.
31. Zhai, S., & Smith, B. (1999). Multi-Stream Input: An Experimental Study of Document Scrolling Methods. *IBM Systems Journal*, *38*(4), p. 642-651.