Precise Selection Techniques for Multi-Touch Screens

Hrvoje Benko
Department of Computer Science
Columbia University
New York, NY
benko@cs.columbia.edu

Andrew D. Wilson, Patrick Baudisch
Microsoft Research
One Microsoft Way
Redmond, WA
{awilson, baudisch}@microsoft.com

ABSTRACT
The size of human fingers and the lack of sensing precision can make precise touch screen interactions difficult. We present a set of five techniques, called *Dual Finger Selections*, which leverage the recent development of multi-touch sensitive displays to help users select very small targets. These techniques facilitate pixel-accurate targeting by adjusting the control-display ratio with a secondary finger while the primary finger controls the movement of the cursor. We also contribute a "clicking" technique, called *SimPress*, which reduces motion errors during clicking and allows us to simulate a hover state on devices unable to sense proximity. We implemented our techniques on a multi-touch tabletop prototype that offers computer vision-based tracking. In our formal user study, we tested the performance of our three most promising techniques (*Stretch*, *X-Menu*, and *Slider*) against our baseline (*Offset*), on four target sizes and three input noise levels. All three chosen techniques outperformed the control technique in terms of error rate reduction and were preferred by our participants, with *Stretch* being the overall performance and preference winner.

Author Keywords
Touch screens, tabletop displays, two-finger, bi-manual, interaction techniques, precise target acquisition.

ACM Classification Keywords
H.5.2 [Information Interfaces and Presentation]: User Interfaces — Input devices and strategies, Interaction styles, Evaluation/methodology.

INTRODUCTION
The ability to directly touch and manipulate data on the screen without using any intermediary devices has a very strong appeal to users. In particular, novices benefit most from the directness of touch screen displays. A fast learning curve and inherent robustness (no movable parts) makes touch screens an ideal interface for interacting with public installations, such as information kiosks, automated teller machines, ticketing machines, or gambling devices.

CHI 2006, April 22–28, 2006, Montréal, Québec, Canada.

Figure 1. Precise dual finger selection techniques enable pixel-precise selections and manipulations on multi-touch screens. This image shows the use of Dual Finger X-Menu in selecting "slow 10X" mode.

While touch screen use is widespread in special purpose applications, the slow adoption of touch screens into more general computing devices has be attributed to known issues of relatively high error rates, arm fatigue, and lack of precision [2]. Due to technical restrictions, most commercially available touch screen devices in use today are only capable of tracking a single point on the surface of the device. However, with the recent emergence of many multi-touch prototype devices [1, 11, 14, 22, 25, 26], research on multi-finger and multi-hand touch interactions has increased [4, 7, 27]. In addition to dealing with the same issues as the single-touch machines, the underlying technology of multi-touch sensitive devices (such as vision-based sensing [25, 26]) often tends to make their input more noisy.

When running software developed for a normal mouse interface on such a touch screen, these issues become problematic. Today's WIMP (windows, icons, menus and pointing) user interfaces require frequent selection of very small targets. For example, window resize handles are often just 4 pixels wide. Noisy input, lower tracking resolution, and large potential touch area of a finger now become a problem. Furthermore, fingertips can occlude small targets depriving users of visual feedback during target acquisition.

Also, the user's hands and arms may contribute to the occlusion problem. Depending on screen orientation, the user may be forced to either look "under hand" (with horizontally positioned screens) or "over hand" (with angled or vertically positioned screens). Finally, it is often difficult to decide the optimal point in the finger's contact area which should anchor the cursor, leaving the usual choice to the center of mass. This can lead to a small but pervasive disconnect between the user's expectations regarding cursor position and what is actually being sensed and computed.

These issues have been recognized by researchers who have proposed several solutions: adding a fixed cursor offset [21], enlarging the target area [19], and providing on-screen widgets to aid in selection [2]. Unlike previous work, we explore the benefits of multi-touch capable devices to provide fluid dual-finger interactions for pixel-accurate targeting. In the techniques presented in this paper, the secondary (non-pointing) finger can quickly modify or switch cursor manipulation modes without disrupting the primary (pointing) finger.

We present an iteratively designed set of five techniques that allow the user to simultaneously perform both cursor steering and selection of assistance mode (in the form of cursor offset, scale, speed reduction, or a combination). In addition to the precise selection techniques, we contribute a "clicking" technique, called *SimPress*, which reduces motion errors during clicking and allows us to simulate a hover state on devices unable to sense proximity.

RELATED WORK

Difficulties with precise interactions on touch screen devices have been addressed before, initially by Potter et al. [21]. Their *Take-Off* technique provides a cursor with a fixed offset above the tip of a finger when the user is touching the screen. Lifting the finger off the screen triggered selection ("click"). While this method is effective for most targets sizes, it has been found ineffective when the target size is smaller than 4 pixels. Sears and Shneiderman [24] explored cursor stabilization improvements that effectively slowed down the cursor movement in various regions around the initial finger contact point, thus allowing for pixel-precise selection. While this method performed well for the target acquisition task, a precise steering task, such as drawing, would be hard due to varying cursor speed.

More recently, Albinsson and Zhai [2] explored several on-screen widgets for increasing precision while selecting small targets on a touch screen. Their interactions were designed to be used with touch screens capable of reporting only a single contact point and therefore the users were required to execute multiple discrete steps before selecting the target. These steps were delimited by the user lifting their finger from the screen, thus impeding the overall interaction performance. Interestingly, they observed that even though their baseline zooming technique (*ZoomPointing*) performed best out of the four techniques compared, its main drawback of losing overview or context can be a significant problem in many applications.

Increasing the relative size of screen targets has also been explored by scaling the display space [19] or scaling the motor space [3, 6]. The work of Olwal and Feiner [19] experimented with hand gestures that activated various levels of fish-eye distortion in the interface to facilitate target selection. Techniques that adaptively increase the motor space while leaving the displayed image unchanged, such as those by Blanch et al. [6] or Baudisch et al. [3], show promising results without introducing screen distortions, but require that the system know all target locations. This information might not be available in many of today's applications. More importantly, such techniques require the use of a relative pointing device such as a mouse. Without such devices, they introduce an unpredictable cursor offset when applied directly to an absolute pointing device such as a touch screen.

Buxton [8] identified that most current user interfaces require an interaction model consisting of at least 3 different states (out-of-range, tracking, and dragging). However, many touch sensitive devices can only reliably sense location in one state thus making it hard to disambiguate between dragging and tracking (hover).

The use of a stylus (pen) is generally preferred in many interfaces that require precise interactions. However, while a stylus has a much smaller tip, the associated issues with hand tremor and resolution make the selection task of small targets more difficult than with a mouse. Ren and Moriya [23] report that a limiting size for stylus targets is about 1.8 mm, below which even the stylus interaction requires additional assistance.

Much research has been performed on bimanual interaction in user interfaces. In their pioneering work, Buxton and Myers [9] demonstrated that users tend to parallelize interaction tasks between hands thus gaining significant performance improvements. Bier et al. [5], in their *Toolglass and Magic Lenses* system, allowed the user to control the transparent tool palette with the non-dominant hand, while the dominant hand controlled the primary cursor with the mouse. This simultaneous bimanual operation eliminated many inefficiencies typical of modal interfaces. Some research by Kabbash et al. [16] points in the opposite direction, claiming that requiring the user to coordinate actions of their hands in order to perform two-handed interactions may complicate the overall task and slow performance.

Two-finger and two-handed interactions for the activation of various tools, menus and widgets have been explored by researchers in many related fields: on tabletop surfaces [7, 12, 22, 27], 3D volumetric displays [13], tangible user interfaces [15, 20], virtual reality [10], and augmented reality [4]. Rekimoto presented a novel capacitance-based sensing architecture, called *SmartSkin* [22], together with

several multi-handed and multi-touch techniques for enabling the manipulation of objects projected on the surface. Cutler et al. [10] present bimanual interaction research where the user can perform two-handed interactions to manipulate both their perspective and three-dimensional models on a 3D tabletop display. Recently, Malik et al. [18] explored using vision-based hand tracking over a tabletop surface to perform multi-finger and whole-hand gestures to interact with a remote display.

DESIGN GUIDELINES

To address the precision problem of touch screen interactions we developed Dual Finger Selections, two finger (or bi-manual) interactions that allow the user to improve targeting precision when necessary without hindering simple targeting tasks. Dual Finger Selections were designed in an iterative fashion. During their development we followed the following guidelines:

1) *Keep simple things simple:* The ability to directly touch an object in order to select it (without any offset or displacement) is probably the most appealing aspect of touch screens. We aim to support this direct manner of interaction and require that further assistance is invoked only when the user explicitly requests it.
2) *Provide an offset to the cursor when so desired:* In addition to not occluding the cursor with the finger while pointing, a spatial offset between the finger and the cursor makes it possible to obtain a more comfortable position when pointing to hard to reach areas (for example: the corner of the screen). The offset should be user-invoked and temporary. Fixed permanent offsets (such as the one used in the *Take-Off* technique [21]) require the users to continuously compensate their targeting even in situations when the target is large enough to be easily selected by direct touch.
3) *Enable the user to modify the control-display ratio:* Provide an increased control-display ratio when so desired to aid in targeting and to reduce tracking noise. This change of the control-display ratio should not involve the pointing finger.

Before discussing the details of our dual finger selection techniques, it is important to outline the device requirements that enable our interactions.

ENABLING TECHNOLOGIES

Our techniques require a multi-touch screen that is capable of simultaneously tracking at least two independent contacts on the surface. We also assume that in addition to the location of contacts, their contact areas are reported as well. A brief description of our prototype multi-touch device can be found later in this paper.

Since most touch screens and tabletops cannot identify which of the individual user's fingers or hands is touching the surface without special gloves [7], we have assumed that the first contact with a tabletop surface is a *primary finger*, while the second contact is a *secondary finger*. The primary finger is the finger that the user normally points with and tends to be the index finger on the dominant hand. The secondary finger is a helper finger which in Dual Finger Selections can be any other finger on the same or opposite hand. In most cases we observed that users used the index finger on their non-dominant hand as the secondary finger. With some interactions, a single-handed operation is desired, and then the thumb of the dominant hand can serve as a secondary finger.

SimPress Clicking

In addition to disambiguating between fingers, our interactions require that the device implement a clicking operation distinct from merely coming in contact with the surface. Previous techniques that address this issue, such as *Land-On* or *Take-Off* [21], implicitly perform a "click" when the contact between their finger and the surface is either established or broken, respectively. Such techniques provide a mechanism for clicking, but do not address the needs of current user interfaces that require at least 3 different interaction states [8]: *out-of-range*, *tracking* (also known as hover or proximity), and *dragging*. Both tracking and dragging states require the contact position to be continuously reported; however, most current touch-sensitive devices only sense location when the contact is actually touching the surface, making it difficult to approximate those two states. A possible solution is to use pressure-sensing technology and map the increased pressure to a dragging state, and light pressure to a tracking state.

Since our device does not report pressure directly, we simulated a pressure-sensitive device by mapping the changes in the finger's contact area to the changes in pressure. In addition to applying different finger areas to different pressure states (which has been implemented in the Synaptics touchpad devices as described by MacKenzie et al. [17]), we have attempted to reduce cursor noise while the user is changing the pressure states (clicking). The stabilization of the cursor movement during clicking is a crucial aspect of our technique, which we call *SimPress (**Sim**ulated **Press**ure)*. SimPress requires the user to apply a small rocking motion with their finger in order to perform a

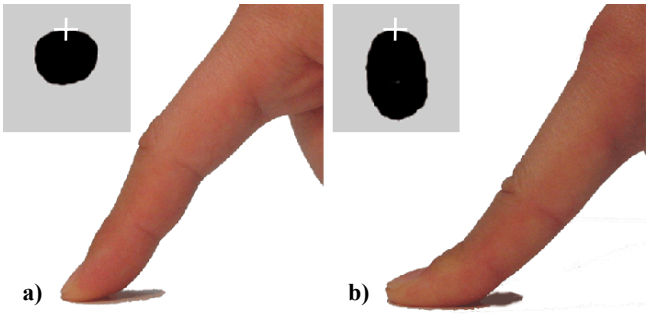

Figure 2. A small rocking motion of the user's finger triggers the *SimPress* clicking technique: a) tracking (hover) state, b) dragging (click) state. (The top left corners show the actual area of contact detected by our device as well as the stabilized cursor location.)

"click", as seen in Figure 2. Since the user starts pointing with their finger tip and then rocks the finger to click, the increase in area happens predominately in one direction: from the tip point towards the user's wrist. We used this fact to stabilize the cursor position by fixing the cursor location to the top middle point of the contact area, rather then the center of mass (Figure 2). In our preliminary experiments, we found that this point naturally moves much less than the center point and therefore reduces the cursor noise during clicking.

By fixing the cursor to the top-middle point, the user is also able to make a more drastic change in the contact area without significantly disturbing the cursor location, which aids in reduction of the unintentional clicks. Two thresholds on contact area were established to disable spurious switching between the clicking states due to noise or hand tremor. Crossing the high threshold activates the click-on state, and crossing the low threshold returns back to click-off state. Due to the finger size differences, these high and low thresholds should be automatically recalibrated for each person. Currently, the calibration is done manually.

SimPress only works if the user is always approaching the tabletop from the same direction, otherwise the orientation of the hand and arm has to be taken into account. A future improvement can potentially use the orientation of the click itself to track the orientation of the user's hand. However, given that in our experiments, the orientation of the user interface was fixed, our users tended to orient themselves straight-ahead. In our dual finger selection techniques, all click events are always triggered by the primary finger.

DUAL FINGER SELECTIONS

Out of the five design prototypes, the first two present simple two-finger extensions of the current state of the art. However, those provide important starting points for our later designs and serve as baseline techniques for comparisons. Therefore, we include them in our discussion.

Dual Finger Offset

Our initial and simplest Dual Finger Selection technique, called *Dual Finger Offset*, provides a user triggered cursor offset. The cursor offset is not enabled by default. However, by placing a secondary finger anywhere on the surface, the cursor is subsequently offset with respect to the primary finger by predefined fixed amount. This offset always places the cursor above the primary finger. To accommodate both left- and right-handed users the cursor is placed to the left or to the right of the primary finger based on the relative position of the secondary finger. For example, by placing the secondary finger to the left of the secondary finger to the left of the primary, the cursor appears to the left of and above the primary finger.

Dual Finger Midpoint

To provide both variable offset and enable finer control of the cursor speed, we have designed the *Dual Finger Midpoint* technique. This technique is triggered by placing the secondary finger on the surface. The cursor is then offset to the midpoint between the primary and the secondary finger. A similar behavior occurs on any resistive touchpad that places the pointer at the midpoint of all touches (e.g., SMART Board Interactive Whiteboard [1]).

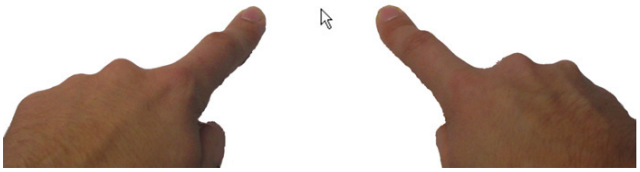

Figure 3. Dual Finger Midpoint technique positions the cursor at exactly the halfway point between the two fingers, giving the user both a cursor offset as well as a variable reduction of cursor speed.

While both fingers are in contact, moving either or both fingers controls the movement of the cursor. Clicking is still performed only by the primary finger. This technique allows for variable reductions in cursor speed: when both fingers are moving in the same direction and the same speed, the cursor follows with the same speed, while when only one finger is moving, the cursor moves with half the speed of that finger.

While the Dual Finger Midpoint technique was very appealing to our initial testers and very simple to master, it did not provide enough assistance for selecting the smallest targets (2 pixels or less). At best, this method reduces the finger speed by a factor of 2 which yields good results for most targets; but it does not provide enough control for the smallest targets. An additional shortcoming of this technique is that not all locations on the screen are equally accessible. For example, screen corners are not accessible using midpoint selection. Consequently, the utility of this technique is somewhat limited by the fact that in today's user interfaces small targets often are located in the corners of the screen.

Dual Finger Stretch

Inspired by the strong performance of *ZoomPointing* technique [2], we designed a *Dual Finger Stretch* technique that allows the user to adaptively scale a portion of the screen with the secondary finger while the primary finger performs the selection. To allow for simultaneous "stretching" and selection, the primary finger provides the initial anchor location around which the user interface is scaled, while the secondary finger identifies the corner of the square area which will be scaled. By moving the secondary finger closer or further away from the primary finger, the square stretching area is reduced or expanded as illustrated in Figure 4. Lifting the secondary finger from the table resets the interface to its default un-stretched state. Upon this reset, the cursor is offset with respect to the primary finger and is placed where it was located in the stretched state. The cursor offset is reset when all fingers are removed from the table. The extent of control-display ratio manipulation depends on two physical limits: the

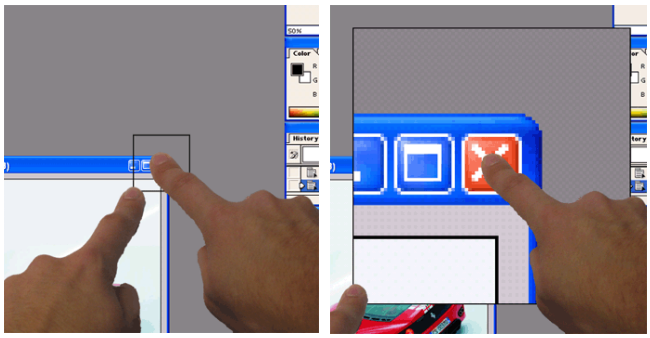

Figure 4. *Dual Finger Stretch* technique adaptively scales the user interface: a) The secondary finger specifies the square zooming area centered at the primary finger's location, b) Primary finger performs precise selection while, simultaneously, the secondary finger adjusts the level of magnification.

closest perceptible distance between user's fingers and the largest diagonal of the screen. For most common mid-screen manipulations, Dual Finger Stretch enables control-display ratios roughly up to 10. By allowing clutching and repeated zooming, it may be possible to further increase this ratio.

The *Dual Finger Stretch* technique has several advantages over the *ZoomPointing* technique primarily due to the dual finger design. First, zooming and selection are not decoupled into two separate actions. Instead they can happen concurrently which results in a fluid interaction. Second, the interface scales in all directions from the original primary finger's location. This provides an important advantage over traditional rectangle selection where the two points specify the diagonal corners of the zooming rectangle (also known as *bounding box zoom*). With the rectangle selection, the user tends to place the primary finger off target in order to "capture" the target in the zoomed area, while with Dual Finger Stretch, the user places the primary finger directly on target and the interfaces scales underneath in all directions. Placing the finger off-target requires the user's primary finger to traverse an increased distance to perform final selection because the target will appear to move away from the finger as the zoom level increases. By encouraging placement of the primary finger as close to the target as possible, the eventual distance that this finger will need to traverse to acquire the target is minimized.

Dual Finger X-Menu

To allow users to adaptively adjust the control-display ratio as well as obtain cursor offset while looking at an un-zoomed user interface, we have designed the *Dual Finger X-Menu* widget. This circular menu is invoked whenever the secondary finger establishes contact with the surface. It is positioned so that the finger is located at its center. The user can select a particular assistance mode by moving the secondary finger to any of the desired regions of the menu (Figure 5). Dual Finger X-Menu has six selection areas

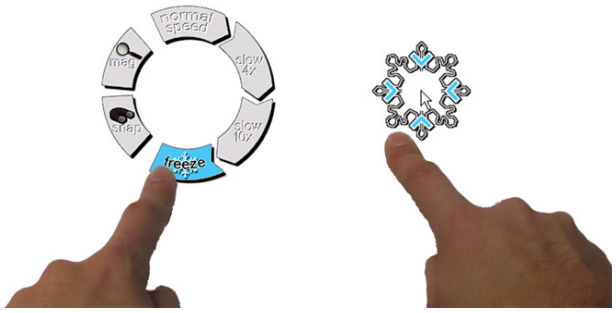

Figure 5. Dual Finger X-Menu enables the user to adjust the cursor speed by crossing over a particular area of the on-screen menu. Freeze mode is selected, making the cursor completely immobile.

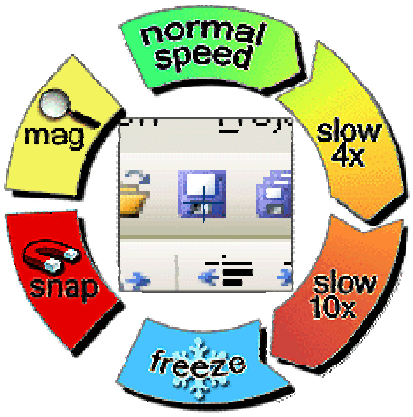

Figure 6. Dual Finger X-Menu contains four selection areas for cursor speed control (normal, slow 4x, slow 10x and freeze), and two toggle areas (snap and magnify). Magnify mode presents an integrated magnification widget in the middle of the menu, while Snap mode removes the current cursor offset.

shown in Figure 6. Four areas control the relative speed of the cursor: *normal*, *slow 4X*, *slow 10X*, and *freeze*. Normal mode moves the cursor with the same speed as the primary finger; the two slow modes reduce the speed of the cursor by a factor of 4 and 10 respectively, while freeze mode "freezes" the cursor in place, disabling any cursor movement.

In preliminary experiments, we found that the ability to completely stop the cursor from moving has two benefits. First, by freezing the cursor, the user can quickly and easily establish a desired cursor offset. This is accomplished by freezing the cursor temporarily, moving the finger to achieve the desired offset, and then unfreezing the cursor again. Second, when selecting very small targets, even small amounts of noise can cause an error. Such noise can be due to device tracking errors, tremor in the user's hand, or noise due to the clicking motion. By freezing the cursor in place, the user can ensure that the desired selection is successful even in very noisy conditions.

The left two areas on the crossing menu invoke two helper modes: "snap" and "magnify". When snapping is triggered, the cursor offset (if any) is removed and the cursor snaps

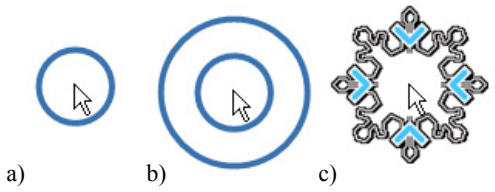

Figure 7. Cursor notification widget signals the current amount of cursor speed reduction: a) 4x reduction, b) 10x reduction, and c) frozen cursor.

back to the current location of the primary finger. This mode is useful in repositioning the cursor in the slow movement modes because it is easy to run out of tracked screen space when using the slow cursor modes. Magnify mode presents a small magnification area in the middle of the crossing menu that shows the enlarged area under the cursor. The magnification factor is fixed at 2X. This mode is particularly useful when the primary finger overlaps the cursor. In this case the magnified image acts as a lens showing the portion of the interface obstructed by the primary finger. A simple cursor notification widget displays which cursor speed level is currently selected, without requiring the user to refer back to the menu. The behavior of this notification widget can be seen in Figure 7.

Dual Finger X-Menu is not operated by clicking, but rather by "crossing" the finger into a particular area, which enables more experienced users to activate modes by simply performing quick strokes in a particular direction. With practice, this selection can be made without looking, and could therefore allow for an expert mode in which the menu could be completely hidden from the user. Removing the secondary finger from the surface will cause the menu to disappear.

Dual Finger Slider

Encouraged by the possibilities of the different interaction modes of Dual Finger X-Menu and the simplicity of Dual Finger Midpoint, we developed the *Dual Finger Slider* technique, which incorporates the menu's most useful features, but simplifies and streamlines the overall interaction (Figure 8). Given that two finger interactions are a very natural way of specifying distance, we have designed this interaction using the distance between fingers to switch between cursor speed reduction modes. This technique does not present an on-screen widget to the user. Instead, it relies completely on the user's ability to gauge the spatial relationship between their fingers. The same cursor notification widget (Figure 7) is used to signal the cursor speed to the user.

Moving the secondary finger towards the primary finger reduces the cursor speed in 3 discrete steps. This allows for the same reductions in cursor speed that is available in Dual Finger X-Menu: *normal*, *slow 4X*, *slow 10X*, and *freeze*. Moving the secondary finger away from the primary increases the speed up to the normal speed. Continuing to

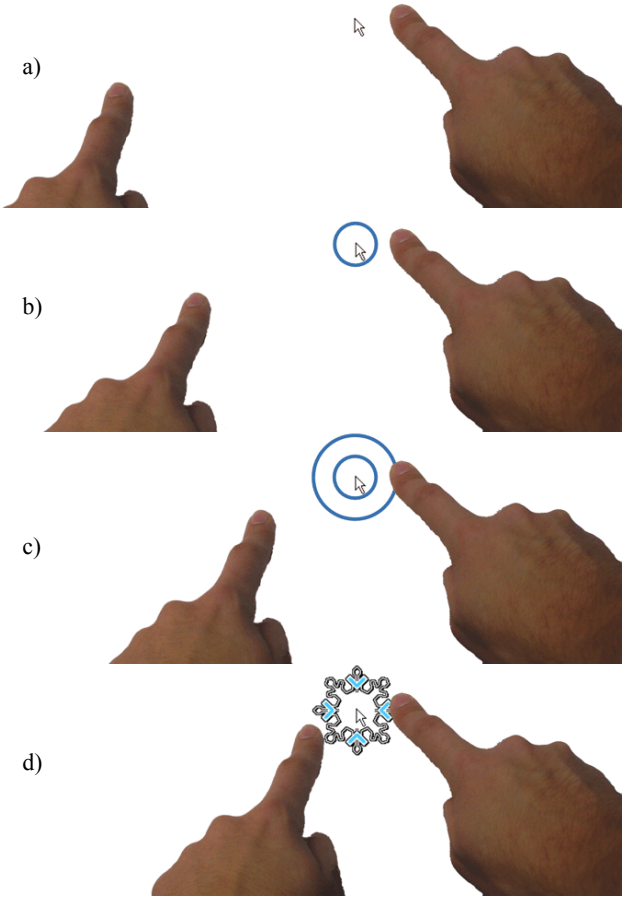

Figure 8. Dual Finger Slider – the right finger (primary) controls the cursor, the left finger (secondary) is invoking the invisible slider; speed reductions modes are achieved by moving the fingers closer together: a) normal, b) slow 4x, c) slow 10x, d) frozen cursor mode.

move the fingers apart triggers a "snap" which warps the cursor back to the primary finger's location. Snapping is signaled by a distinct sound effect. The distance that the secondary finger traverses in switching speed reduction modes is predefined and is not dependent on the distance between the fingers. The modes are remembered even after the user lifts the secondary finger which allows for clutching in the interaction.

PROTOTYPE MULTI-TOUCH TABLETOP DISPLAY

Our interaction techniques are designed for present and future multi-touch screens, and to some extent, they are device independent. However, we have developed them on a prototype multi-touch tabletop display to facilitate research and user studies (see Figure 9). The prototype uses a diffuse screen coupled with an infrared camera and computer vision algorithm to detect contacts on the tabletop surface. The screen is back-projected with the projector integrated in the base of the table below the screen. Our display uses the infra-red light spectrum for contact detection while all projection is done in the visible

Figure 9. Our multi-touch back-projected tabletop display prototype uses an infra-red illuminant and camera to detect contacts through a diffuse surface.

spectrum. This separation allows the computer vision algorithms to ignore the projected display in order to see only surface contacts. A similar approach is used in the TouchLight [26] display system and elsewhere.

The screen resolution of our prototype multi-touch device is 1024 x 768 (pixels), which, given the screen dimensions of 61 x 46 (cm), yields a pixel size of 0.6mm. The finger that is about 1.5 cm wide covers about 25 screen pixels.

LABORATORY USER STUDY

To evaluate the Dual Finger Selection and SimPress techniques, we conducted a user study that challenged the users to select small and large targets using the various techniques. Additionally, we were interested in how well these techniques perform on devices of very low precision. Such devices include touch screens based on a small number of sensing elements, touch screens based on noisy computer vision processes, and many prototype research systems which do not achieve the precision of the mouse (e.g. see [25, 26]). Accordingly, in our experiments we added synthetic noise to the sensing system described above, and systematically varied its magnitude.

Twelve paid participants (9 male and 3 female), ages 20–40, participated in the experiment. All subjects were frequent computer users. They had varying experience with the touch screens, ranging from "monthly" use to "several times a day", with the average corresponding to "weekly" use. All subjects used their right hand as their dominant hand. Eleven subjects identified themselves as right-handed. The single left-handed subject preferred using their right hand for mouse operation and chose to use the right hand as the dominant pointing hand in the experiments. The subjects were pre-screened for color blindness.

The subjects were asked to perform a simple reciprocal target selection task, with square targets of varying widths, separated by a fixed distance of 100 pixels. This task is loosely based on the Fitts' Law target acquisition task, but without the variation of distance. The task involved clicking on a green square target that was surrounded by a green circle. The other (inactive) target was colored red and the targets alternated between trials. The users were instructed to click on the current green target as fast and as accurately as possible. We recorded both movement times and error rates, but we analyzed completion times only for successfully completed trials. We had hypothesized that the smallest targets might not be reliably selectable by all the techniques tested and therefore were more interested in the impact of our techniques on the reduction of error rate, than the completion time.

The experiment consisted of two parts: an evaluation of the SimPress technique and a comparative evaluation of the four dual finger selection techniques under varying amounts of noise. Both used the same testing infrastructure to present targets to the user, measure user performance and log all experimental data. In addition, the users completed a post-experiment user preference questionnaire.

Part One: SimPress Clicking

We wanted to determine the performance of SimPress clicking technique to obtain a baseline measure of the minimal target width that is possible to select reliably without additional assistance. An additional motivation was to ensure that our subjects mastered and were comfortable using SimPress, since we required them to use it throughout later experiments. Our subjects were first given an introduction to the SimPress technique and then allowed to perform 1 practice run before the actual experiment.

A within-subjects, repeated measures design was used consisting of 5 target widths (1, 2, 4, 8, and 16 pixels). The widths were chosen to represent the range of smallest available targets in a typical GUI. For example, the smaller toolbar buttons tend to be between 15 and 20 pixels wide, while the resize handles are sometimes less than 5 pixels wide. The experiment consisted of 5 sets (1 set per width) of 5 trials each, for a total of 25 trials per user. The order of the sets was randomized across users.

Our hypothesis was that the users would be able to reliably select only the largest of our small targets (16 pixels) and that the finger's occlusion of the target and the small amount of noise still present while clicking would make the selection of other target sizes difficult.

Results

We performed a repeated measures ANOVA on the mean error rate data and found the significant main effect with target width ($F_{(4,44)}=62.598$, $p<0.001$). The data are summarized in Figure 10. Paired samples t-tests show no significant differences between the user's performance with 8 and 16 pixel targets. A significance difference in performance is shown between 2 and 4 pixel targets ($t_{(11)}=3.95$, $p=0.002$) and 4 and 8 pixel targets ($t_{(11)}=4.16$, $p=0.002$). The difference between 1 and 2 pixels is of borderline significance ($t_{(11)}=2.41$, $p=0.034$).

Contrary to our hypothesis, we found that the threshold target size, below which the *SimPress* technique is not

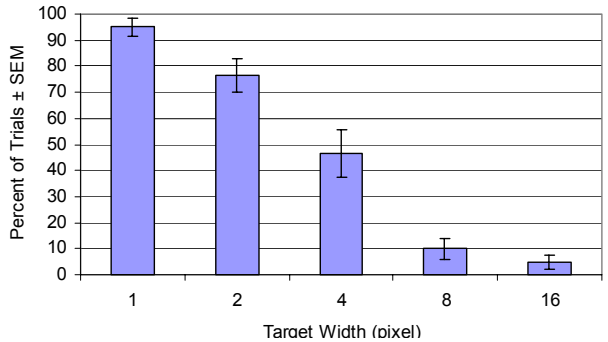

Figure 10. Mean error rate (%) using the SimPress technique alone without any assistance.

reliable alone, is around 8 pixels. These results show that *SimPress* is a viable option for use for most general selection tasks in the current user interface.

Part Two: Comparison of Four Dual Finger Techniques

For the second part of the experiment, we tested the performance of our Dual Finger Selection by comparing the three most versatile techniques (Stretch, X-Menu and Slider) and the Offset technique. By providing no additional assistance other than the cursor offset, the Offset technique served as a baseline. Even though the Midpoint technique received very positive response from our initial testers, this technique was not included due to the relatively small assistance that it offered in selection (the maximum reduction of the cursor speed was a factor of 2) and lack of equal accessibility to all screen locations.

Additionally, we were interested in how our techniques would perform in the presence of noisy input. We note that many touch screen devices provide noisier input than standard relative pointing devices such as a mouse. This is particularly true of a whole class of touch screen devices that depend on the video signal for their touch recognition. In addition to noise in the video stream, such devices often require that the video signal is up-sampled to match the screen's resolution. This up-sampling introduces additional sampling noise. In order to test how our techniques deal with increased noise, we added Gaussian noise to the position of each tracked finger, creating three noise levels: low (no additional noise), medium (Gaussian noise with σ=0.5), and high (Gaussian noise with σ=2).

While the noise can be reduced with a use of a filter (Kalman filter being the most commonly used), this solution either results in a slight cursor lag or overshoot when the finger's velocity abruptly changes, as is the case with any start or stop of the finger. We believe that there is a benefit to having interaction techniques that adaptively allow the user to reduce the noise when so desired, leaving the noisy, but lag-free, input otherwise. By manipulating the control/display ratio, Stretch, X-Menu, and Slider implicitly allowed the reduction of the input noise as well.

Our study followed a within subjects design that tested 3 noise levels, 4 techniques, and 4 target widths (1, 2, 4, and 8 pixels) per block. Within each block, the user performed 6 trials resulting in a total of 288 trials per user. To eliminate the effects of switching selection strategies (for example deciding to use a different cursor speed reduction for a different target size) we discarded the first trial in each block. All our techniques were modified to completely reset after each click in order to ensure the same starting state for all trials.

Our main hypothesis was that techniques that increase the control/display ratio lessen the impact of the input noise. Therefore, Stretch, X-Menu and Slider should be less affected by the increase in noise, than the Offset technique. The second hypothesis was that Slider would perform better than X-Menu since the Slider is controlled by the natural bi-manual way of specifying spatial extent (finger distance), rather that the independent finger actions in X-Menu.

Results

We performed a 3 (Noise) x 4 (Technique) x 4 (Width) repeated measures ANOVA on the mean error rate data and found the significant main effects across all conditions. As expected, noise had a significant main effect on the error rate ($F_{(2,22)}$=20.24, $p<0.001$). This confirmed that more errors were committed in the higher noise levels. Significant main effects were also present for width ($F_{(3,33)}$=150.4, $p<0.001$) and technique ($F_{(3,33)}$=169.138, $p<0.001$). Paired samples t-tests show that the Offset technique created significantly more errors than the rest ($t_{(11)}$=14.298, $p<0.001$), while Stretch was better than the X-Menu or Slider ($t_{(11)}$=2.864, $p=0.015$). No significant difference was found in the error rate between X-Menu and Slider techniques.

The interaction of technique and width ($F_{(9,99)}$=29.473, $p<0.001$, Figure 11) is interesting as it shows that our assistive techniques (Slider, X-Menu, and Stretch) all performed exceptionally well (less than 5% error rate) in all noise conditions for targets 2 pixels or larger (no statistical differences between techniques). For the smallest target (1 pixel), Stretch outperformed X-Menu and Slider (with borderline significance $t_{(11)}$=2.64, $p=0.023$). The interaction of noise and technique was also significant ($F_{(6,66)}$=8.025, $p<0.001$, Figure 12). While the increase of noise greatly degraded performance of the Offset technique, the other 3 techniques show no statistically significant effects to the various noise levels. This confirmed our main hypothesis

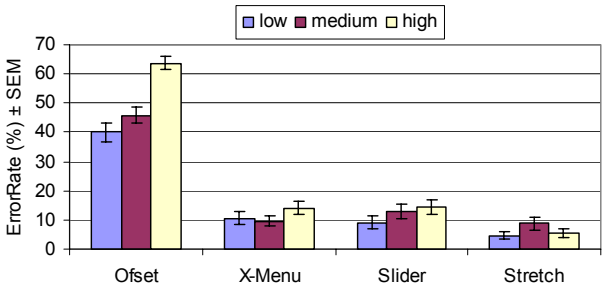

Figure 11. Interaction of Noise x Technique graph for error rate (%).

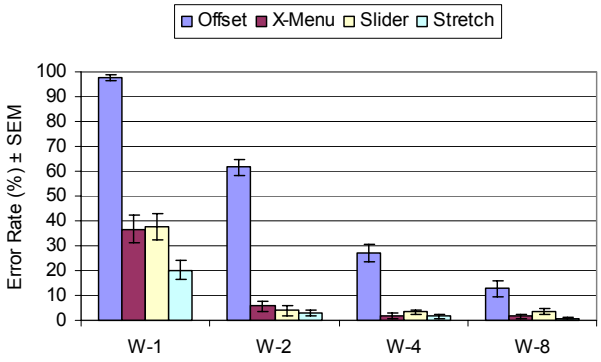

Figure 12. Interaction of Technique x Width graph for error rate (%).

that users are able to lessen the impact of noise and low precision by using techniques that allow for control-display ratio adjustments.

Due to the dramatically high number of errors committed by our users using the Offset technique, our data contains several blocks without a single successful trial (all in 1 pixel width condition). While this prevented us from performing a repeated measures ANOVA on movement times, we present the informal analysis of median movement time values for blocks for which we have data. Median times were chosen to correct for the typical skewing due to reaction time data. This also removed the influence of any outliers in the data. Aggregating the movement times across all noise levels and all target widths, the Stretch technique is on average 1s faster than Slider ($t_{(11)}$=5.011, p<0.001). There do not appear to be statistical differences in the performance times of Slider and X-Menu. This failed to confirm our second hypothesis that Slider would outperform X-Menu. Offset's performance times were comparable to other techniques, indicating that users did not believe that spending more time on targeting would yield more precise targeting with Offset technique. Figure 13 shows the performance of techniques with respect to target width. The data shows a general trend of more time being spent on targeting smaller targets.

Subjective Evaluation

The users filled out a post-experiment questionnaire rating their experience with four techniques on a 5 point Likert scale (1 being most negative and 5 being most positive) They were asked to comment on the following categories: mental effort, learning time, hand fatigue, enjoyment, and performance in low, medium and high conditions.

Overall, techniques received significantly different results ($F_{(3,33)}$=45.9, p<0.001). X-Menu required the most mental effort (average score of 2.88), and the longest time to learn (average score of 2.09). Data shows no significant statistical differences between techniques with respect to hand fatigue. Stretching was the most enjoyable (average score of 4.12), followed closely by Slider technique (average score of 4.08). We also asked users to rate their overall preference for the technique for selecting small targets.

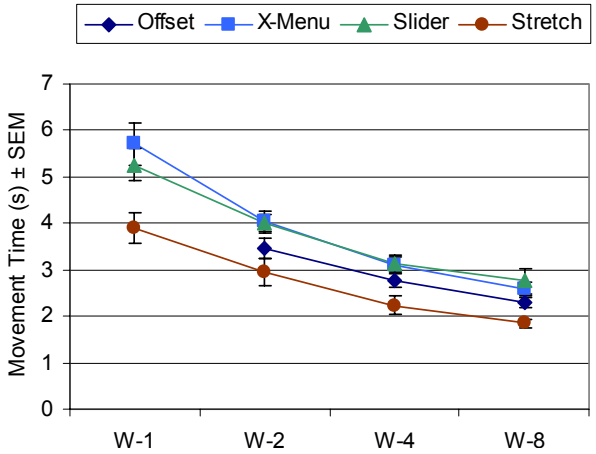

Figure 13. Mean performance time (s) with respect to target widths. (Notice: time data is not shown for Offset technique at 1 pixel due to lack of successfully completed trials.)

Stretch was the most preferred (7 subjects), followed by Slider (4 subjects), while only one user preferred X-Menu.

DISCUSSION AND CONCLUSIONS

Out of the four compared techniques, the top performer and most preferred technique, Stretch, was the only one that did not provide a cursor offset. This clearly demonstrated that the benefit of increased target size successfully compensated for the fingertip occlusion factor. The data from this experiment is consistent with the results from a study by Albinsson and Zhai [2] which also showed that their baseline zooming technique outperformed on-screen widgets that provided cursor speed control.

We feel that Dual Finger Stretch is a simple and powerful interaction that utilizes the distance between fingers in a very natural way. However, in many applications, scaling may have an undesired effect of losing overview of the interface. Therefore, we were very pleased with the strong performance of Slider and X-Menu, which provided comparable error rates with a small time penalty of about 1s when compared to Stretch. In addition, as our subjects' written comments point out, those techniques strongly benefit by the ability to freeze the cursor. As one subject describes, freezing the cursor is a functional equivalent to presenting a user-controlled "are you sure?" dialogue for clicking which enables the user to select a particular point without risk of mistake, or go back and re-target. This was particularly useful with higher noise levels. Experience should also substantially improve our Slider and X-Menu because zooming is a very familiar interaction for most users. As such, it might have an unfair advantage when compared to other speed-controlling techniques.

The SimPress clicking technique exceeded our performance expectations. This enables the novice users to reliably click on targets that are as small as 8 pixels. We believe that with practice and more thorough calibration, this threshold could be further reduced. Some future work on stabilization is needed in order to completely remove the remaining noise

from clicking. An additional SimPress modification was implemented, but not tested, permitting the user to rapidly click on targets without requiring the rocking motion. This timer solution generates a click event if the contact was present on the surface for less than 0.4 s. This allowed the simple selection of large targets to remain as direct as possible while more complex interactions, such as drag and drop, can be performed using the SimPress technique.

Our study results show that *Dual Finger Selections* present viable solutions for increasing precision and accuracy in a small target selection task. They are designed to be used on most multi-touch screens, perform well with the increase of input noise, and fully utilize the benefits of dual finger interactions. Overall, these techniques provide a palette of interactions from which the user may chose depending on the application.

ACKNOWLEDGMENTS

We would like to thank Ed Cutrell, Ken Hinckley, and Steven Feiner, for their support and helpful comments.

REFERENCES

1. SMART Technologies, http://www.smarttech.com.
2. Albinsson, P.A. and Zhai, S. High Precision Touch Screen Interaction. *Proc. CHI '03*, 2003, pp. 105-112.
3. Baudisch, P., Cutrell, E., Hinckley, K. and Eversole, A. Snap-and-go: Helping Users Align Objects Without the Modality of Traditional Snapping. *Proc. CHI '05*, 2005, ACM Press, pp. 301-310.
4. Benko, H., Ishak, E. and Feiner, S. Cross-Dimensional Gestural Interaction Techniques for Hybrid Immersive Environments. *Proc. IEEE VR '05*, 2005, pp. 209-216.
5. Bier, E.A., Stone, M.C., Pier, K., Buxton, W. and DeRose, T. Toolglass and Magic Lenses: The See-Through Interface. *Proc. ACM SIGGRAPH '93*, 1993, pp. 73–80.
6. Blanch, R., Guiard, Y. and Beaudouin-Lafon, M. Semantic Pointing: Improving Target Acquisition with Control-Display Ratio Adaptation. *Proc. CHI '04*, 2004, ACM Press, pp. 519-526.
7. Butler, C.G. and Amant, R.S. HabilisDraw DT: A Bimanual Tool-Based Direct Manipulation Drawing Environment. *Proc. CHI '04 Extended Abstracts*, 2004, pp. 1301-1304.
8. Buxton, W. Issues and Techniques in Touch-Sensitive Tablet Input. *Proc. ACM SIGGRAPH '85*, 1985, pp. 215-224.
9. Buxton, W. and Myers, B. A Study in Two-Handed Input. *Proc. CHI '86*, 1986, pp. 321-326.
10. Cutler, L.D., Fröhlich, B. and Hanrahan, P. Two-Handed Direct Manipulation on the Responsive Workbench. *Proc. Symposium on Interactive 3D Graphics*, 1997, pp. 107-114.
11. Dietz, P. and Lehigh, D. DiamondTouch: a Multi-User Touch Technology. *Proc. UIST '01*, 2001, pp. 219–226.
12. Forlines, C. and Shen, C. DTLens: Multi-User Tabletop Spatial Data Exploration. *Proc. UIST '05*, 2005, ACM Press, pp. 119-122.
13. Grossman, T., Wigdor, D. and Balakrishnan, R. Multi-Finger Gestural Interaction with 3D Volumetric Displays. *Proc. UIST '04*, 2004, pp. 61-70.
14. Han, J.Y. Low-Cost Multi-touch Sensing Through Frustrated Total Internal Reflection. *Proc. UIST '05*, 2005, ACM Press, pp. 115-118.
15. Hinckley, K., Pausch, R., Proffitt, D. and Kassell, N.F. Two-Handed Virtual Manipulation. *ACM Transactions on Computer-Human Interaction*, *5* (3). pp. 260-302.
16. Kabbash, P., Buxton, W. and Sellen, A. Two-Handed Input in a Compound Task. *Proc. CHI '94*, 1994, ACM Press, pp. 417-423.
17. MacKenzie, I.S. and Oniszczak, A. A Comparison of Three Selection Techniques for Touchpads. *Proc. CHI '98*, 1998, ACM Press, pp. 336-343.
18. Malik, S., Ranjan, A. and Balakrishnan, R. Interacting with Large Displays from a Distance with Vision-Tracked Multi-Finger Gestural Input. *Proc. UIST '05*, 2005, ACM Press, pp. 43-52.
19. Olwal, A. and Feiner, S. Rubbing the Fisheye: Precise Touch-Screen Interaction with Gestures and Fisheye Views. *Conference Supplement of UIST '03*. pp. 83-84.
20. Patten, J., Ishii, H., Hines, J. and Pangaro, G. Sensetable: A Wireless Object Tracking Platform for Tangible User Interfaces. *Proc. CHI '01*, 2001, pp. 253-260.
21. Potter, R.L., Weldon, L.J. and Shneiderman, B. Improving the Accuracy of Touchscreens: An Experimental Evaluation of Three Strategies. *Proc. Proc. CHI '88*, 1988, pp. 27-32.
22. Rekimoto, J. SmartSkin: an Infrastructure for Free-hand Manipulation on Interactive Surfaces. *Proc. CHI '02*, 2002, pp. 113–120.
23. Ren, X. and Moriya, S. Improving Selection Performance on Pen-Based Systems: A Study of Pen-Input Interaction for Selection Tasks. *ACM Transactions on Computer Human Interaction 7*(3). pp. 384-416.
24. Sears, A. and Shneiderman, B. High Precision Touchscreens: Design Strategies and Comparisons with a Mouse. *International Journal of Man-Machine Studies*, *43* (4). pp. 593-613.
25. Wilson, A. PlayAnywhere: A Compact Tabletop Computer Vision System. *Proc. UIST '05*, 2005, ACM Press, pp. 83-92.
26. Wilson, A. TouchLight: An Imaging Touch Screen and Display for Gesture-Based Interaction. *Proc. ICMI '04*, 2004, pp. 69-76.
27. Wu, M. and Balakrishnan, R. Multi-Finger and Whole Hand Gestural Interaction Techniques for Multi-User Tabletop Displays. *Proc. UIST '03*, 2003, pp. 193-202.