TouchCuts and TouchZoom: Enhanced Target Selection for Touch Displays using Finger Proximity Sensing

Xing-Dong Yang[1,2], Tovi Grossman[1], Pourang Irani[3], George Fitzmaurice[1]

[1]Autodesk Research
210 King St. East, Toronto,
ON, M5A 1J7 Canada
{firstname.lastname}@autodesk.com

[2]Dept. of Computing Science
University of Alberta, Edmonton,
AB, T6G 2E8 Canada
xingdong@cs.ualberta.ca

[3]Dept. of Computer Science
University of Manitoba, Winnipeg,
MB, R3T 2N2, Canada
lastname@cs.umanitoba.ca

ABSTRACT

Although touch-screen laptops are increasing in popularity, users still do not comfortably rely on touch in these environments, as current software interfaces were not designed for being used by the finger. In this paper, we first demonstrate the benefits of using touch as a complementary input modality along with the keyboard and mouse or touchpad in a laptop setting. To alleviate the frustration users experience with touch, we then design two techniques, *TouchCuts*, a single target expansion technique, and *TouchZoom*, a multiple target expansion technique. Both techniques facilitate the selection of small icons, by detecting the finger proximity above the display surface, and expanding the target as the finger approaches. In a controlled evaluation, we show that our techniques improve performance in comparison to both the computer mouse and a baseline touch-based target acquisition technique. We conclude by discussing other application scenarios that our techniques support.

Author Keywords

Touch input, target expansion.

ACM Classification Keywords

H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

General terms

Design, Human Factors.

INTRODUCTION

Touch input is often considered intuitive and effective [7]. It is becoming a major input modality, especially on mobile devices. Recently, manufacturers have started producing laptops equipped with touch-screens, allowing users to use their fingers to directly interact with their applications. However, legacy applications typically utilize a ribbon or tool palette, consisting of small tiled icons, and research has shown that such targets are difficult to select with the finger due to occlusion [25] and accuracy [19] problems.

CHI 2011, May 7–12, 2011, Vancouver, BC, Canada.

While there is an abundance of work in creating new UI paradigms specifically for touch [17, 26], it is unlikely that the legacy applications used most often by users would ever go through such a transformation. Very limited work has looked at making general purpose UIs more touch friendly, without impacting their non-touch experience.

Ideally, software interfaces should seamlessly adapt to the input modality [4, 8]. With the use of proximity based sensing [5, 10, 16, 22], UI components could transition to a touch-optimized variant only when the user's hand approaches, and thus be left unaltered for mouse input. We propose and study the benefit of this idea by introducing two new techniques: *TouchZoom* (Figure 1) and *TouchCuts*. Both techniques utilize target expansion as a basic UI enhancement to facilitate finger input on interfaces that host many small and tiled icons, such as ribbons. Both techniques only activate when a finger approaches, and therefore do not affect traditional cursor input.

In the following sections, we first perform a study investigating the benefits of touch input in a laptop configuration. Motivated by the high error rates found for small targets, we describe our new techniques, *TouchCut*s, a single target expansion technique, and *TouchZoom*, a multiple target expansion technique. A controlled study shows that our techniques improve performance in comparison to both the computer mouse and a baseline touch-based target acquisition technique, Shift [25]. We conclude by discussing other application scenarios that our techniques could support.

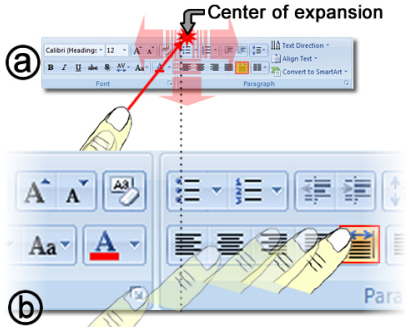

Figure 1 - *TouchZoom*. (a) The ribbon expands at the intersection of a finger's motion vector and the top edge of the ribbon. (b) After the expansion, finger movement is adjusted to acquire the new position of the highlighted goal target icon.

RELATED WORK

Selection with Touch

Having finger input available in tandem with the mouse in a shared UI environment allows these devices to complement each other, thus providing numerous benefits. Researchers have found that the finger can be faster than a mouse when selecting targets greater than 3.2mm [24]. Nevertheless users can perform better with a mouse than with a finger in tasks requiring fine movement and selection [15] as also confirmed by Forlines et al. [7].

Clearly, the size of a UI component will affect touch input. Established guidelines suggest that the target size for touch should be greater than 10mm [19]. However, smaller icons (e.g. 5mm) are still common in most of the current user interfaces. Finger based interactions also suffer from occlusion problems. Proposed solutions to alleviate this concern include the Offset Cursor [20], that relocates the cursor 0.5" above the finger; DTMouse, which presents the cursor between two fingers [6], Shift, which dynamically places a copy of the occluded area in a shifted callout [25], and Dual-Finger-Selections, which supports precise selections on multitouch displays [1].

In Shift, users slide the finger contact on the display, to fine tune a crosshair cursor position and select the target using a take-off gesture. Shift's callout was designed to be triggered only when necessary. However, in most cases, the selection can be separated into 2 steps – invoking the callout and adjusting the cursor to make a selection. Escape [26], allows selecting small and tiled targets by assigning a unique directional gesture for selection. An experimental study showed that Escape could perform 30% faster than Shift on targets smaller than 5mm while still maintaining a similar error rate. Similarly, Sliding widgets [17] require users to slide on a target in a particular direction for selection. These two techniques require changing the traditional UIs to visualize directional clues, which may not be suitable for an interface also shared with traditional cursor input.

Mouse-based Target Expansion

Target expansion techniques for the mouse have also been widely researched [3, 13, 27]. These techniques have been found helpful for selecting a single target, by expanding the target as the cursor approaches. However, in real-world situations, targets are often laid out in a tiled arrangement. The existing approaches for expanding tiled targets include expanding a single item that is predicted to be the desired target [27, 14] or expanding the predicted item as well as its immediate neighbors [23]. Obviously, one of the limitations of these methods is the lack of guaranteeing that the desired target will be magnified due to prediction errors. Note that when using a mouse the user can still make a precise selection on an unexpanded target [23]. However with finger input, the user's intended target may be too difficult to select if it is not expanded.

Zhai et al. [27] suggest that if space permits it, all the targets in the view should be expanded. However, it is typical for ribbons or toolbars to span the entire display space, so there would not be room to expand all targets. An alternative would be to use some form of fisheye distortion [2], but these have been shown to be harmful to pointing and selection tasks [9, 27].

Touch-Based Target Expansion

Despite its popularity in the target acquisition literature, target expansion has not been widely explored for touch. Olwal et al. [18] proposed using a rubbing gesture or a second finger tap to 'zoom into' a small target before attempting to make a selection. Similarly, Benko et al. [1]'s two-handed techniques also attempt to enlarge the target to ease selection. These techniques have been proven helpful for selecting small targets using bare fingers. However, they have a common limitation that users need to explicitly indicate a target of interest and then expand it manually.

The more traditional, automatic target expansion, has not been explored for touch. This is likely because target expansion relies on tracking the approach of the cursor, or in this case, the finger. This information is typically not available, as most touch devices today do not sense hover information. However, advanced sensing technology has pushed the detection of finger motion at a practical height above the surface [11, 22]. Major manufacturers (Mitsubishi, Primesense, and Cypress) have already announced such types of commercial systems or prototypes [5, 16, 21]. All of these have made target expansion feasible on small touchscreen devices, e.g. laptops.

In summary, target expansion for touch has not been investigated thoroughly, and is promising in that it may improve touch selection without impacting the user interface for traditional cursor input. However, even in the mouse-based research, there are challenges surrounding target expansion yet to be addressed, such as how to apply expansion in tiled-target environments. In the next section, we first demonstrate the added value of finger input in a keyboard/mouse setting. We then present our designs and studies of our techniques.

EXPERIMENT 1: EVLAUATION OF TOUCH

We believe touch input could be particular useful in a laptop configuration, since a mouse may not be available, and the hands, when in a resting state on the keyboard are already quite close to the display. However, despite the prevalence of touch-based laptops, the efficiency of using touch on such devices has not been investigated and is thus not fully understood. To better understand if and when target expansion techniques would be useful, we first study traditional target acquisition in a laptop configuration. While previous work has compared touch to other forms of input [7, 15, 24], here, on screen target acquisition is unique, since the hand would be moving from a horizontal plane of the keyboard to a vertical plane of the display. In

addition, strictly following the KLM GOMS model, we would predict that such a task would require homing time (switching to the touch input device) and pointing time (acquiring the target). We postulate that one of the primary benefits of using touch for target selection tasks is allowing pointing and homing to take place in parallel. We are unaware of any investigation into this issue.

Motivated to answer these open questions, the goal of this experiment is to evaluate the performance of touch versus a mouse and touchpad in a task involving the use of a pointing device in conjunction with a keyboard.

Apparatus

Our study was conducted on a Dell SX2210 21.5" touch-screen monitor, as it provided more accurate touch input then current touch-enabled laptops. The display was used in conjunction with a standard desktop keyboard. To simulate a laptop configuration, we lowered the height of the touch-screen so that the button of the display is 4 cm above the keyboard, which is about the same distance that can be found on a Dell TouchSmart TX2 tablet PC. The display was tilted to a comfortable viewing angle (13° to the vertical plane). A USB touchpad (97 × 78 mm) from Ergonomic was placed below the space bar of the keyboard, and was raised to the same height as the keyboard (Figure 2).

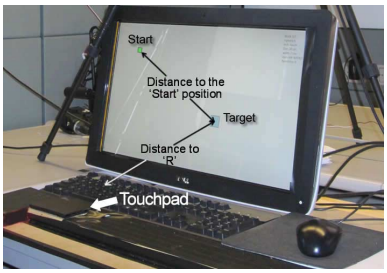

Figure 2 – Hardware setup for Experiment 1.

Participants

Twelve paid participants (7 males and 5 females) between the ages of 21 and 56 participated in this study. All participants were right-handed. They were all familiar with a computer mouse and touchpad, and had previous experience with mobile touch-screen devices.

Task and procedure

The task required participants to click a key on the keyboard and then using the same hand select a square target of various sizes in a random location on the screen. This task is analogous to that frequently employed by users of text editing programs, where the main task is typing on a keyboard but requires users to switch to a mouse or a touchpad to click an icon or other on-screen widget. The right hand was used for the mouse and touchpad. The left hand (non-dominant for all participants) was used for touch, since we felt testing touch with the dominant hand may be biased, since in some cases users may want to use the non-dominant hand (for example, if a target is on the left side, or if their right hand is on the mouse). Participants were told to only use their index finger during the touch conditions.

In the mouse and touchpad conditions, participants were asked to position a cursor inside a 0.5×0.5cm 'Start' square prior to the start of a trial. A trial started after participants pressed a keyboard key and finished after a target was successfully selected. Ideally, we would have like to use the 'F' key when using touch and the 'J' key when using the mouse and touchpad, as these are the typical resting keys for the left and right hands respectively. However, we used the 'R' key for touch and the '\' key for mouse, and the 'J' key for touchpad since the distance between these keys and their respective input devices more closely matched the distances on a Dell TouchSmart TX2 tablet PC.

Participants were asked to finish the task as fast and as accurately as possible. They were encouraged to take breaks during the experiment, which lasted about 40 minutes.

Design

The experiment employed a 3×4×3 within-subject factorial design. The independent variables were *Pointing Device* (*Finger, Mouse, and Touchpad*); *Target Distance* (*18, 24, 30*, and *36*cm)*;* and *Target Size* (*0.5, 1*, and *2*cm).

The size of the target was chosen to be close to the size of the icons in real-world applications. For instance, 0.5cm is approximately the same size as the ***Bold*** button in Microsoft Word. Similarly, 1cm is approximately the same size as the *Paste* button. Target distance was measured from the center of the goal target to the center of the 'Start' position in the mouse and touchpad conditions. Distance was measured from the center of the 'R' key to the center of the goal target in the touch condition. Target locations were the same for all conditions. The 'Start' square was repositioned (in the mouse and touchpad conditions) according to the target position to ensure it satisfied the distance condition.

Windows cursor acceleration was turned on to facilitate pointing by using the cursor. If the user missed the target, they had to click again until successful. In each trial, participants performed tasks in one of each *Pointing Device* × *Target Distance* × *Target Size* combination. The experiment consisted of 5 blocks, each consisting of 3 repetitions of trials. The first block was used as practice trails, thus the data was not used in analysis. The *Pointing Device* was counter balanced among participants. The *Target Distance* and *Target Size* were randomized among trials.

Results and discussion

Dependent measures included the number of errors and the average task completion time. This data was analyzed using Repeated-measures ANOVA and Bonferroni corrections for pair-wise comparisons.

Task completion Time

ANOVA yielded a significant effect of *Pointing Device* ($F_{2,22}$ = 14.43, p < 0.001), *Target Distance* ($F_{3,33}$ = 3.61, p < 0.05), and *Target Size* ($F_{2,22}$ = 101.72, p < 0.001). There is also significant interaction effects on *Input Device* × *Target Distance* ($F_{6,66}$ = 8.68, p < 0.001), *Input Device* × *Target Width* ($F_{4,44}$ = 29.07, p < 0.001), and *Target Distance* × *Target Width* ($F_{6,66}$ = 5.53, p < 0.001). The interaction effects were mainly caused by the poor performance of finger touch on the target of 0.5cm (see Figure 3).

Overall (including trials with errors), the performance of finger (1528ms) was significantly faster than mouse (1639ms) (p < 0.05), which was significantly faster than touchpad (2242ms) (p < 0.001). As expected target size has more impact on finger than mouse or touchpad (Figure 3 left). Participants spent more time selecting the smallest target using finger than using mouse or touchpad.

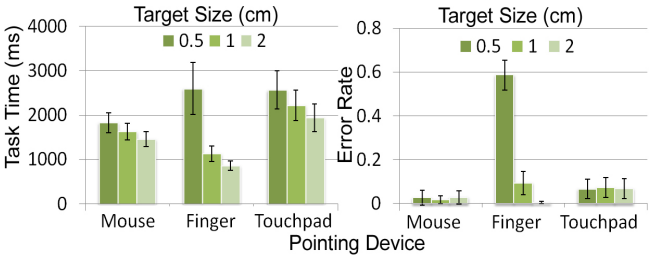

Figure 3 – Task time and error rate shown by Technique and Target Size. (Error Bars show 95% CI in all figures)

Number of errors

ANOVA yielded a significant effect of *Input Device* ($F_{2,22}$ = 57.46, p < 0.001), *Target Distance* ($F_{3,33}$ = 7.68, p < 0.05), and *Target Size* ($F_{2,22}$ = 202.36, p < 0.001). We also found significant interaction effects on *Input Device* × *Target Distance* ($F_{6,66}$ = 4.18, p = 0.001) and *Input Device* × *Target Width* ($F_{4,44}$ = 225.43, p < 0.001).

Overall, touch made significantly more errors (23%) than touchpad (7%) (p < 0.001), which made significantly more errors than mouse (2%) (p < 0.001). Figure 3 right shows that participants made closed to 60% errors using touch on target size 0.5cm but made less errors than mouse on target of size 2cm.

Fitts' Law Analysis

To perform a Fitts' Law analysis, we removed all trials in which errors occurred. Linear regression tests indicated that the task highly conformed to Fitts' Law, for all three conditions, showing that touch is constantly faster than mouse or touchpad across all index of difficulties (Figure 4). It can be seen that the main difference is due to the 'a' constant, which is typically reaction time, but for this study, encompasses the homing time as well. This is an interesting results, as it shows that touch does allow homing and pointing to be carried out concurrently. To repeat the analysis without homing time, we subtracted the elapsed time until the cursor began to move in the mouse and touchpad conditions. After doing so, we still see a 16% advantage of touch over the mouse (p < 0.001), and a 35% advantage over the touchpad (p < 0.001).

Summary

The study demonstrates certain benefits of using touch in a routine task, which requires users to switch from a keyboard to a pointing device prior to start acquiring a target. For targets that are at least 1cm large, touch was 36% faster than the mouse and 52% faster than the touchpad. Even without the homing time, touch was 8% faster than the mouse and 35% faster than the touchpad. However, as expected, the performance of touch decreased significantly with small targets. In particular, our study shows that touch completely fails for target sizes of 0.5×0.5cm. Unfortunately, many graphical user interfaces contain targets of this size, so further considerations must be made for touch to be practical on desktop applications.

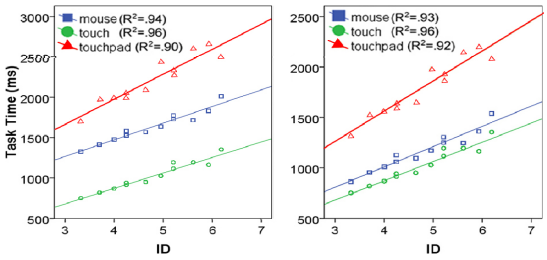

Figure 4 –Task completion time with homing time (left) and without homing time (right) by the index of difficulty.

EXPANDING TARGET TECHNIQUES FOR TOUCH

The results of Experiment 1 provide an important lesson: if targets are big enough, using touch to acquire them can have significant advantages. However, in desktop applications, increasing the size of the targets is not practical, as it would diminish the experience for users who never intend to use touch. It is also impractical to expect traditional legacy applications to be rewritten specifically for touch. Instead, we propose that **UI components transition to be optimized for touch *only* when the user intends to use touch**. Already, numerous technologies exist to detect the proximity of fingers [5, 11, 16, 21, 22]. Expanding the targets as a finger approaches the screen could be an efficient mechanism to overcome the challenges encountered in Experiment 1. In this section, we initiate the investigation of expanding targets for touch, through the design of two techniques: *TouchCut*s and *TouchZoom*.

TouchCuts

TouchCuts are a basic implementation of expanding targets for touch, where only certain targets within a UI palette, such as a toolbar or ribbon, expand when the finger approaches it. Only this subset of targets is accessible through touch, as surrounding targets may become occluded by them. As such, *TouchCuts* are akin to keyboard shortcuts, providing efficient access to some of an application's commands. However, they are not exhaustive or do not provide

a replacement for other command access methods. *TouchCuts* use a visual gloss overlay, to indicate to the users which of the targets are accessible through touch. We also implemented a simple method to allow users to customize which targets are *TouchCuts*. The user taps close to a desired target using their finger, and drags the mouse to specify the size of a TouchCut (Figure 5). Similarly, tapping on a TouchCut, followed by a mouse click on the enlarged button can remove it. Using a transparent overlay window, we were able to prototype an application independent implementation of *TouchCuts*, that could be used to customize and access *TouchCuts* on any windows program.

Figure 5 –Left: define a *TouchCuts*. Right: *TouchCuts* expands when a finger approaches.

The advantage of this technique is that it is a single target expansion, so no target prediction is required, and no targets are displaced. The limitation is that it cannot provide access to every target within a tool palette. Therefore, *TouchCuts* are most suitable for functions that are used frequently. In the case when a function is not available through *TouchCuts*, the user will have to use a mouse or re-customize the *TouchCuts*.

TouchZoom

We also wanted to develop a technique that could provide full access to a palette's icons. We focus our design on the ribbon, because it has become a common UI component for desktop applications, but the technique would work for any horizontal tool palette. Our approach is to magnify the entire ribbon, with a center of expansion at a predicted endpoint, similar to what Zhai et al. previously proposed [27]. This ensures expansion of the desired target, even if there is an error in the endpoint prediction. Thus, unlike *TouchCuts*, the *TouchZoom* makes every target within the ribbon accessible through touch. One concern, which will require investigation, is that the goal target may become offset from its original location, and in the worst case, the target may be displaced off-screen.

For prediction, a 2D motion vector is generated based on the projection of the finger movement on the screen. We use only the last 2 points of the sample to estimate the intended vector, as it tended to perform better than other regression algorithms we implemented. The intersection of the motion vector and the top of the ribbon determines the center of expansion (*CE*), and the ribbon expands once the index finger crosses an expansion point in a sufficient speed. If *CE* falls outside of the ribbon, it will be placed on the corresponding ribbon endpoint. The *CE* was fixed at the top of the ribbon in all cases. When the index finger crosses a certain distance threshold, the *CE* is calculated and the resulting ribbon expansion occurs (Figure 1). When the prediction is wrong, the goal target will be offset from its original location at a magnitude proportional to the prediction error. To recover from an off-screen error, users can either re-launch the expansion by moving below the distance threshold, or scrub the ribbon with a touch gesture to pan the target back into view.

We anticipated that the accuracy of the prediction would be dependent on the time when the ribbon expansion is triggered. Expanding late might lead to better prediction since the finger would be closer to its goal. However, considering that no prediction algorithm works perfectly [12, 13, 14], a user will need a certain amount of time to perceive target displacement, and to adjust his/her finger motion accordingly. Thus, it may be preferable to expand the ribbon early.

Another design option we considered is to group sets of ribbon icons together, and to set the *CE* to the center of the predicted group (For groups adjacent to the screen edge, the *CE* would be set to the edge) (Figure 6). This would allow a user to aim his/her finger movement at the group containing the desired icon, instead of aiming at a desired icon itself. Having this larger initial target would minimize prediction errors, and, once expanded, the user could adjust his/her finger movement to make the final selection. However, the target offset would be proportional to the distance between the target and the center of its group.

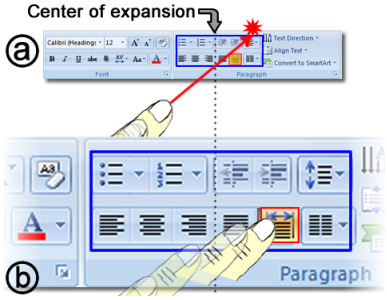

Figure 6 – (a) The ribbon expands at the center of a group of icons shown in the blue outline. (b) User adjusts finger movement to the new position of the highlighted target icon.

In the following section, we study the relevant parameters for our *TouchZoom* design. In Experiment 3 we will compare *TouchZoom* to *TouchCuts* and two baseline techniques.

EXPERIMENT 2: EVALUATION OF TOUCHZOOM

In this study, we were interested in measuring the impact of the pertinent design parameters for *TouchZoom*, to optimize the techniques efficiency. The parameters we investigated were group size, target position, and expansion point.

Apparatus

The hardware setup was the same as in Experiment 1. In addition, we used four OptiTrack motion capture cameras to capture the off-screen movement of the user's index finger. The cameras have 100Hz frame rates and millimeter accuracy. We put 3 reflective markers on a user's hand (Figure 7). The marker on the user's index finger was tilted to the right side to avoid occlusion. This should be consid-

ered enabling technology only, simulating the more practical technologies discussed earlier.

Figure 7 – Reflective marker placement.

Participants

Ten paid participants (5 males and 5 females) between the ages of 20 and 34 participated in this study. All participants were right-handed. They were all familiar with graphical user interfaces, and had previous experience with mobile touch-screen devices.

Task and Procedure

The task required participants to use their left index finger to press the 'O' key on a keyboard and then select a target button in our abstracted ribbon using the same finger. Upon pressing the 'O', the participants were instructed to select the target as fast and as accurately as possible. A trial started after the 'O' was pressed and finished after a target was successfully selected. Participants were encouraged to take breaks during the experiment. The entire experiment lasted about 45 minutes.

Design

The ribbon was 1.5cm high and 45cm width, and was rendered near the top of the screen in a window which has the same width as the ribbon and the same height as the screen. A 100ms animation was used for the Ribbon expansion. In each trial, a 0.5×0.5cm goal target was shown in the ribbon. An expansion magnification level of 3x, which was based on Experiment 1, was used to reduce the chance of errors. The ribbon had 90 icons across, and 3 icons in each column. The 'O' key was centered with the ribbon.

The experiment employed a 5×2×2×9 within-subject factorial design. The independent variables are *Group Size* (*1*, *3*, *9*, *15*, and *30*); *Expansion Point* (*Late* and *Early*)*; Target Y Position* (*Up* and *Down*), and *Target X Position* (*1* to *9*).

Group Size (GS) – Group size indicates the number of icons in a row that a group has. We chose to explore a range of groups size, that evenly divided into the 90 targets across: 1, 3, 9, 15, and 30.

Groups were visualized using vertical bars (Figure 8). For targets close to the screen edges, the participants were shown that by biasing their acquisition movement towards the edge of the screen, off-screen errors could be minimized.

Expansion Point (EP) – The distance to the goal target was measured as the distance from the 'O' key to the target in 3D space. Expansion Point took on the values 90% and 40%, representing the amount of distance travelled to the targets, before the expansion occurred. We chose 40% as a minimum value because our informal test showed that the distances below 40% could significantly impair the prediction. The value of 90% was chosen as it has been suggested by previous expanding target techniques [13].

Target Y Position (TY) –In the *Up* condition, the target was placed on the top row. In the Down condition, the target was placed in the bottom row.

Target X Position (TX) – In each trial, the target was placed in one of 9 different horizontal positions across the ribbon (see Figure 8). In addition to the 9 absolute positions, we were also interested in investigating the effects of 3 relative positions (left, middle, and right) of a target within a group of the ribbon. To ensure each group size had exactly 3 targets in each of these relative positions, we slightly shifted some of the X positions by ±1 icon

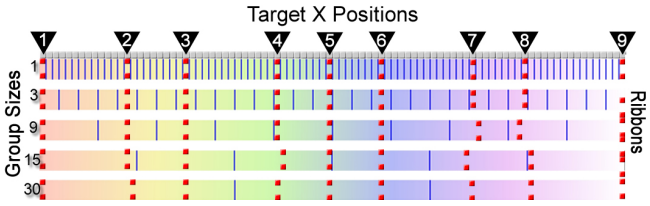

Figure 8 – Illustration of target positions (red dots) within each of the 5 group sizes. The gradient effect was added to provide spatial grounding. Blue bars indicate group borders.

The experiment consisted of 3 blocks, each consisting of 1 trial for each combination of *Group Size (GS)* × *Expansion Point (EP)* × *Target Y Position (TY)* × *Target X Position (TX)*. The order of *Group Size* was randomized between subjects. Within each group size, *Expansion Point* was randomized. Finally, target position was randomized in each *Group Size* × *Expansion Point* pair.

Dependent measures included the number of off-screen errors, the number of selection errors, and the average task completion time. An off-screen error was recorded when a target was pushed into off-screen space. A selection error was recorded when a participant missed a target. Task completion time was recorded as the time elapsed from the 'O' being pressed to a successful selection on the target.

Results and Discussion

The results were analyzed using Repeated-measures ANOVA and Bonferroni corrections for pair-wise comparisons. Before the analysis, we checked the ordering effects of group size on all the dependent measures, and found no significant effects.

Task completion time

ANOVA yielded a significant effect of *TX* ($F_{8,72} = 13.65$, $p < 0.001$). Interestingly, we found no significant effect of *GS* ($F_{4,36} = 1.16$, $p = 0.35$), *EP* ($F_{1,9} = 0.9$, $p = 0.37$), and *TY* ($F_{1,9} = 0.67$, $p = 0.44$). There were significant interaction effects on *GS* × *TX* ($F_{32,288} = 7.17$, $p < 0.001$), and *TY* × *EP* ($F_{1,9} = 6.1$, $p < 0.05$).

Post-hoc analysis showed that task time decreased significantly towards the center of the ribbon (Figure 9 left). In par-

ticular, task time at *TX* 1, 7, 8, and 9 were significantly longer than at 3, 4, 5, 6 (all $p < 0.05$).

Task completion time without off-screen error

There was a significant difference between trials when off-screen errors occurred (2700ms s.e. 119.74) and when off-screen errors did not occur (1094ms s.e. 20.82) ($F_{1,9} = 172.89$, $p < 0.001$).

After removing these trials (7%), we found significant effects of *TX* ($F_{8,72} = 12.35$, $p < 0.001$), *EP* ($F_{1,9} = 47.13$, $p < 0.001$), and *TY* ($F_{1,9} = 14.22$, $p < 0.005$). There was a weak effect of *GS* ($F_{4,36} = 2.88$, $p = 0.04$), but pair-wise comparison showed no significant difference between group sizes. There was a significant interaction effect on *GS* × *TY* ($F_{4,36} = 0.67$, $p < 0.05$). Figure 9 (left) shows the task time with and without off-screen errors.

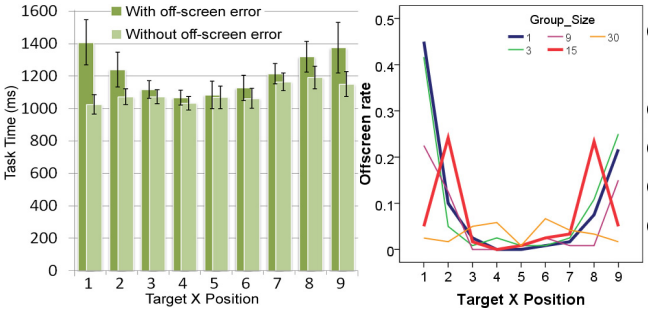

Figure 9 – Left: Task time shown by target position. Right: Off-screen rate shown by target position and group size.

The participants performed the task faster when the ribbon was expanded at 40% distance (1038ms s.e. 25.5ms) than when it was expanded at 90% distance (1147ms, s.e. 18.8). This is because the early expansion allowed users to adjust their initial movement path earlier, instead of following two separate acquisition paths (one for the group, and then one for the goal target position resulting from the expansion).

We also analyzed the effect of the relative position of the targets within their groups, excluding group size 1. We found a significant effect of target position ($F_{1,18} = 14.13$, $p < 0.001$). Targets on the left (1116ms s.e. 24.23ms) and right side (1133ms s.e. 28.62ms) of a group took significantly longer than those on the center of a group (1046ms s.e. 23.68ms). This explains why we found only weak effect of group size. Although big groups (e.g. 15 and 30) have better prediction, their larger target displacements increase acquisition times.

Off-screen Errors

ANOVA yielded a significant effect of *TX* ($F_{8,72} = 22.72$, $p < 0.001$), *GS* ($F_{4,36} = 10.68$, $p < 0.001$), and *EP* ($F_{1,9} = 2.12$, $p < 0.05$). There was no significant effect of *TY* ($F_{1,9} = 0.67$, $p = 0.44$). There were also a significant interaction effect on *EP* × *TX* ($F_{8,72} = 2.58$, $p < 0.05$).

The participants made more off-screen errors when the ribbon was expanded at 40% distance (0.09 s.e. 0.02) than when it was expanded at 90% distance (0.05 s.e. 0.01).

Figure 9 (right) shows that most off-screen errors were made on the targets on the left and right edge of the ribbon (*TX* 1 and 9). Post-hoc analysis showed that big groups (e.g. 15 and 30) introduced significantly less off-screen errors than the smaller groups ($p < 0.05$). It also shows that large group sizes can also cause off-screen errors. For instance, the participants made significantly more off-screen errors on the targets at position *TX* 2 and 8 with group size 15 than with other group sizes ($p < 0.05$). Going back to Figure 8 we see that these two target positions are at the edge of their respective groups. If a participant aimed at the target instead of the group, there was a chance the wrong group would be predicted, pushing the desired group off-screen.

Selection Error

Overall, the average selection error rate was 6.2%. No main effects or interaction effects were found on error rate.

HYBRID INTERPOLATION FOR TOUCHZOOM

Experiment 2 shows that reducing prediction error by increasing the size of the group does not improve the efficiency of the task, because of the larger target offsets. We also found that off-screen errors had an overwhelming effect on overall completion time. In this section we discuss a redesign of *TouchZoom* to prevent off-screen errors.

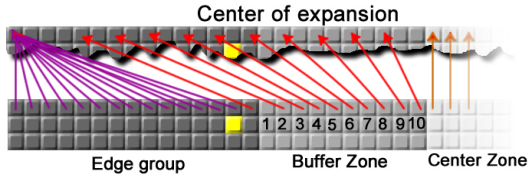

Figure 10 - Illustration of the left side of a ribbon with buffer zone. Red arrows associate the center of expansion for each of the icons in the 3 zones.

As suggested by Experiment 2, larger group sizes are effective in preventing off-screen errors on targets close to the left and right edges of the screen but can be error prone with targets closer to the center. The opposite was true for small group sizes. To leverage the benefits of both designs, we implemented a hybrid technique which uses a group size of 15 on the two edges, and individual ungrouped targets in between. To further reduce the chance of off-screen errors, we use a *buffer* zone between the group of 15 targets, and the center zone. The buffer zone consists of 10 individual targets, but their *CE* is based on a linear interpolation of the screen edge and the *CE* of the first target in the center zone (Figure 10). Having this buffer zone minimizes the impact of a prediction error when aiming at a target in the edge group. The exact sizes of the edge groups and buffer zones were chosen based on prediction error rates from Experiment 2, in an effort to minimize off-screen targets as much as possible.

EXPERIMENT 3

We have described two expanding target techniques for touch, *TouchCuts*, and *TouchZoom*. In study 2 we performed an evaluation of *TouchZoom*, which resulted in a resigned hybrid interpolation. In this study, we measured the perform-

ance of the redesigned *TouchZoom* and *TouchCuts*, in comparison to a baseline input device – the *Mouse* and a baseline touch technique – *Shift*.

While numerous techniques exist for aiding touch-based target acquisition, we used Shift as a baseline, since it does not have any visual impact on the user interface, unless the technique is used. Following previous guidelines, our implementation placed the callout window (16mm in diameter) 22mm on the right side of the initial touch point to facilitate the selection on a ribbon by using the left hand. The callout was placed on the opposite side if the touch point was within 30mm from the right end of the ribbon. We set the escalation time to zero so that the callout popped up as soon as a user touches the screen.

Apparatus

We used the same apparatus as in Experiment 2. Reflective markers were used in the target expansion techniques.

Participants

Twelve paid participants (6 males and 6 females) between the ages of 18 and 34 participated in this study. None of them had participated in the Experiment 1 and 2. All participants were right-handed. They were all familiar with graphical user interfaces. All but one had previous experience with mobile touch-screen devices.

Task and procedure

The participants were asked to press a keyboard key ('\' for the cursor and 'O' for the others), and to select a target (0.5×0.5cm) in a ribbon by using one of the 4 techniques. The task was required to be carried out by using the left hand for all the techniques except for the mouse. In the mouse condition, prior to pressing '\', the participants were asked to place the cursor in a start square (0.5×0.5cm) rendered in the center of the workspace. In the *TouchZoom* condition, we only showed the border between the edge groups and the buffer zone. The buffer zone and the groups of size 1 were invisible to the users. As in Experiment 2, for targets close to the screen edges, the participants were shown that by biasing their acquisition movement towards the edge of the screen, off-screen errors could be minimized. For both target expansion techniques, the expansion point was set to 60%.

Prior to the study, the participants were given a 3 minute training section for each technique. They were encouraged to take breaks during the experiment. The entire experiment lasted about 40 minutes. Participants filled out a post experiment questionnaire upon completion.

Design

The experiment employed a 4×3 within-subject factorial design. The independent variables are *Technique* (*TouchZoom*, *TouchCuts*, *Shift*, and *Mouse Cursor*) and *Target zone* (*Edge Group*, *Buffer Zone* and *Center Zone*).

In each trial, participants performed tasks in one of each *Technique* × *Target zone* combination. The experiment consisted of 3 blocks, each consisting of 30 trials, 10 for each target zone. In each trial, the position of the target was randomized for each target zone. The target was evenly distributed to the left and right side as well as the 3 rows of the ribbon. The order of the presentation of the techniques was counter balanced between participants.

Results

The data was analyzed using Repeated-measures ANOVA and Bonferroni corrections for pair-wise comparisons.

Task completion time

ANOVA yielded a significant effect of *Technique* ($F_{3,33}$ = 245.36, $p < 0.001$) and *Target Zone* ($F_{2,22}$ = 15.82, $p < 0.001$). There was a significant interaction effect on *Input Technique* × *Target Zone* ($F_{6,66}$ = 4.53, $p = 0.001$). Figure 11 left shows average time for *Target Zone* by *Technique*.

Performance with *TouchCuts* (768ms) was faster than *TouchZoom* (1130ms), which was faster than *Mouse cursor* (1280ms) and *Shift* (1883ms). Post-hoc analysis showed significant differences between all pairs of techniques. It is not surprising that *TouchCuts* performed the best overall, since it provides target expansion without any target offset. The difference between *TouchCuts* and *TouchZoom* (362 ms) is the added cost of the target offsets that *TouchZoom* introduces. It is worth reiterating that *TouchCuts* is slightly different from the other techniques, in that it only provides access to a predetermined subset of the ribbon icons.

What was more surprising was the difference between *TouchZoom* and *Shift*. Both techniques require initial and adjustment pointing phases. We believe that *TouchZoom* performed better because with *Shift* the two phases are explicitly sequential, while with the *TouchZoom* technique, the adjustment phase can be predicted and integrated into the end of the initial phase. In addition, *Shift* does not increase the motor activation size of the target.

Performance in the *Edge group* (1306ms) was slightly slower than *Center zone* (1249ms) and *Buffer zone* (1241ms) ($p = 0.001$), while no significant difference was found between *Buffer zone* and *Center zone* ($p = 1$). However, we found no significant effect of *Target Zone* ($F_{2,22}$ = 1.2, $p = 0.321$) in the *TouchZoom* condition, indicating that users can perform equally well across the ribbon.

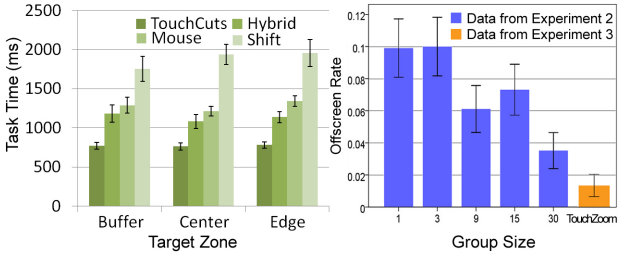

Figure 11 – Left: average task time shown for each target zone by technique. Right: off-screen rate shown by group sizes.

Selection Error

For selection errors, there was a significant effect of *Technique* ($F_{3,33}$ = 8.18, $p < 0.001$), but no significant effect of *Target Zone* ($F_{2,22}$ = 0.03, $p = 0.97$) or interaction effects.

TouchCuts had significantly less errors (0.02 s.e. 0.01) than the *TouchZoom* (0.11 s.e. 0.025), *Mouse cursor* (0.07 s.e. 0.01), and *Shift* (0.08 s.e. 0.02). We found no significant difference between *TouchZoom*, *Mouse cursor*, and *Shift*.

In the *TouchZoom* condition, we also found no significant effect of *Target zone* ($F_{2,22} = 0.07$, $p = 0.93$). This again confirms that users can perform equally well across the ribbon. Average off-screen error rate for the *TouchZoom* condition was 0.014(s.e. 0.006). We also found no significant effect of *Target zone* on off-screen error ($F_{2,22} = 0.836$, $p = 0.45$). This was an encouraging result, demonstrating that the hybrid interpolation we designed based on the results of Experiment 2 effectively minimized the chance of off-screen errors, in comparison to the static group sizes (shown in Figure 11 right). Application designers could color code ribbon or toolbar icons that indicate the zones utilized by the hybrid interpolation.

Subjective Preference

A short questionnaire was administered after the study. All scores reported below are based on a 7-point Likert scale, with 7 indicating highest preference.

The participants gave an average of 6.7 to *TouchCuts* and 5.2 to *TouchZoom* as the two most easy to use techniques. Whereas, *Shift* and *Mouse cursor* received an average of 3.7 and 4.6 respectively. Additionally, the participants gave an average of 6.9 to *TouchCuts* and 5.3 to *TouchZoom* as the two most enjoyable techniques. *Shift* and *Mouse cursor* received an average of 3.9 and 4 respectively. The mouse was rated lower because of both the switching between the keyboard and mouse, and also the small target sizes. Shift was rated lower because of the longer acquisition times. In addition, numerous users reported *Shift* as being fatiguing because it required careful movements of the finger while it was positioned on the screen. When shown a mockup of our techniques in an actual user interface (Microsof PowerPoint), feedback from users was gerneally encouraging. Overall, 83%, 67%, 67%, and 33% of all our participants expressed the desire to use *TouchCuts*, *TouchZoom*, *Mouse*, and *Shift* in the future.

DISCUSSION AND FUTURE WORK

In comparison to a mouse, *TouchCuts* reduced selection times by 40%, but does not provide access to every UI element. It will thus be important to study adoption and performance of TouchCuts in realistic settings, given the potential confusion caused by only some functionality being accessible. Our belief is that this may not be problematic, since traditional hotkeys are similarly only available and used for a subset of commands.

In contrast, *TouchZoom* gives the user access to an entire ribbon, and reduced selection times by 12% in comparison to a mouse. We also know from Experiment 1, that if the user does not have a mouse attached to their laptop, and has to use a touchpad, the levels of improvement would be even more substantial.

It is worth pointing out that we are not trying to replace the mouse. To the contrary, we made our design decisions carefully so that traditional cursor interaction would be unaffected. Note that our techniques are particularly beneficial when frequent mouse-keyboard switching is necessary. If the user is already performing mouse interactions close to the top of the screen, it may make more sense to acquire ribbon icons with the mouse. However, if the user is performing cursor intensive interactions, the non-dominant hand could be used to access UI elements in parallel, saving a round trip of the cursor (for example, changing colors while drawing). While our studies did show that touch can be effective with the non-dominant hand, future studies could explore this form of parallel, bimanual input.

Although the differences were not significant from the mouse or *Shift*, *TouchZoom* did exhibit a higher error rate (11%) in Experiment 3 than we expected. Our observations indicated that some of these errors were caused by users accidently touching the screen at the end of their first ballistic movements, just after the ribbon expanded. Since this error rate was higher than in Experiment 2 (6.2%), one potential explanation is that there were detrimental transfer effects from the *Shift* and *TouchCuts* techniques, where participants could touch the screen imprecisely after an initial ballistic movement.

Our work focused on a laptop configuration for three reasons: Most major manufactures have touch-enabled laptops; the hands are positioned close to the display; a mouse may not be available, and the track-pad is inefficient for pointing tasks. However, our work could generalize to desktop settings as well, but fatigue issues must be considered, since further reaching may be required. In addition, our work assumed proximity sensing was available, and for *TouchZoom*, the control being zoomed is horizontal. In the next section, we discuss several design variations to demonstrate how our techniques could generalize to other scenarios.

Further Design Alternatives

Functionality reduction controls – inspired by *TouchCuts*, we introduce functionality reduction controls to facilitate finger input. A functionality reduction control transitions to a preset touch-optimized component, that has the same screen footprint, but offers a subset of the functionality of its cursor-based counterpart. For example, we implemented a functionality reduction color palette, which replaces the color picker with 12 large color icons when the finger approaches. When users know they want to select one of these main colors, they can do so quickly with the non-dominant hand, saving a cursor round-trip.

Multi-level expansion – In this technique, the ribbon has 2 expansion points: 50% and 80%. It expands half-way if the finger crosses the 50% distance, and fully expands after the finger crosses the 80% distance. No discontinuity will be seen if the finger crosses the 2 expansion point at a sufficient speed. This technique was mainly designed to help users learn to use the *TouchZoom* technique.

Depth-based expansion – Depth-based expansion triggers the expansion when the finger is within a threshold distance to the screen, with the center of expansion equal to the on-screen projection of the current finger position. The finger is first positioned directly above the target of interest, and then the finger moves towards the screen to trigger the expansion. This could be particularly useful for implementing *TouchZoom* on vertical tool palettes, since endpoint prediction would be difficult. In addition, it could be useful on systems with a small proximity sensing range.

Touch activated expansion – To demonstrate the use of *TouchCuts* without proximity sensing, we delay expansion until the finger actually makes contact with the screen. This could also be used for *functionality reduction controls*, if the touch-optimized layout is predictable.

CONCLUSION

We have presented 3 studies to motivate and evaluate our design of *TouchCuts* and *TouchZoom*. We demonstrated that finger input has the benefit of allowing homing and pointing to be carried out concurrently, but suffers from extremely high error rates on icons in existing legacy applications. To support touch in such applications, our techniques trigger a transition of the user interface only when a finger approaches. Thus, controls can be effectively shared by both a traditional cursor and touch. Our study showed positive benefits of both techniques. Furthermore, the results of our studies show the hybrid interpolation used for *TouchZoom* effectively reduces the chance of off-screen errors. Finally, we present several alternative designs to show how our techniques could generalize to scenarios which we did not explicitly study. We believe with the continued increase in popularity of touch-based displays, our techniques may serve as important groundwork for integrating the benefits of touch into existing applications.

REFERENCES

1. Benko, H., Wilson, A., and Baudisch, P. (2006). Precise selection techniques for multi-touch screens. *CHI'06*, 1263-1272.
2. Carpendale, S., Ligh, J., and Pattison, E. (2004). Achieving higher magnification in context. *UIST'04*, 71-80.
3. Cockburn, A. and Brock, P. (2006). Human on-line response to visual and motor target expansion. *GI'06*, 81-87.
4. Carter, S., Hurst, A., Mankoff, J., and Li, J. (2006). Dynamically adapting GUIs to diverse input devices. *ACCESS'06*, 63-70.
5. Cypress Semiconductor Co., http://www.cypress.com/?rID=42793.
6. Esenther, A. and Ryall, K. (2006). Fluid DTMouse: better mouse support for touch-based interactions. *AVI'06*, 112-115.
7. Forlines, C., Wigdor, D., Shen, C., and Balakrishnan, R. (2007). Direct-touch vs. mouse input for tabletop displays. *CHI'07*, 647-656.
8. Gajos, K. and Weld, D. S. (2004). SUPPLE: automatically generating user interfaces. *IUI '04*, 93-100.
9. Gutwin, C. (2002). Improving focus targeting in interactive fisheye views. *CHI'02*, 267-274.
10. Hinckley, K. and Sinclair, M. (1999). Touch-sensing input devices. *CHI'99*, 223-230.
11. Hodges, S., Izadi, S., Butler, A., Rrustemi, A., and Buxton, B. (2007). ThinSight: Versatile Multi-touch Sensingfor Thin Form-factor Displays. *UIST'07*, 259-268.
12. Lank, E., Chun, Y., Cheng, N., and Ruiz, J. (2007). Endpoint prediction using motion kinematics. *CHI'07*, 637-646.
13. McGuffin, M. J. and Balakrishnan, R. (2002). Acquisition of Expanding Targets. *CHI'02*, 57-64.
14. McGuffin, M. J. and Balakrishnan, R. (2005). Fitts' Law and Expanding Targets: Experimental Studies and Designs for User Interfaces. *TOCHI*, 12(4) 388-422.
15. Meyer, S., Cohen, O., and Nilsen, E. (1994). Device comparisons for goal-directed drawing tasks. *Extended Abstracts of the CHI'94*, 251-252.
16. Mitsubishi 3D touch panel, http://techon.nikkeibp.co.jp/ english/NEWS_EN/20090310/166952/.
17. Moscovich, T. (2009). Contact Area Interaction with Sliding Widgets. *UIST'09* 13-22.
18. Olwal, A., Feiner, S., and Heyman, S. (2008). Rubbing and tapping for precise and rapid selection on touch-screen displays. *CHI'08*, 295-304.
19. Parhi, P. Karlson, A. K. and Bederson, B. B. (2006). Target size study for one-handed thumb use on small touch-screen devices. *MobileHci'06*, 203-210.
20. Potter, R., Weldon, L., Shneiderman, B. (1988). Improving the accuracy of touch screens: an experimental evaluation of three strategies. *CHI*'88, 27-32.
21. Primesense Ltd., http://www.primesense.com/?p=486.
22. Rekimoto, J. (2002). SmartSkin: An Infrastructure for Freehand Manipulation on Interactive Surfaces, *CHI'02*, 113 - 120.
23. Ruiz, J. and Lank, E. (2010). Speeding pointing in tiled widgets: understanding the effects of target expansion and misprediction. *IUI'10*, 229-238.
24. Sears, A. and Shneiderman, B. (1991). High precision touch-screens: design strategies and comparisons with a mouse. *IJMMS*, 34(4), 93–613.
25. Vogel, D. and Baudisch, P. (2007). Shift: a technique for operating pen-based interfaces using touch. *CHI*'07, 657-666.
26. Yatani, K., Partridge, K., Bern, M., and Newman, M. (2008). Escape: A target selection technique using visually-cued gestures. *CHI'08*, 285-294.
27. Zhai, S., Conversy, S., Beaudouin-Lafon, M., Guiard, Y. (2003). Human On-Line Response to Target Expansion. *CHI'03*, 177-184.