Shift: A Technique for Operating Pen-Based Interfaces Using Touch

Daniel Vogel
Department of Computer Science
University of Toronto
dvogel@ dgp.toronto.edu

Patrick Baudisch
Microsoft Research
Redmond, WA
baudisch@ microsoft.com

ABSTRACT

Retrieving the stylus of a pen-based device takes time and requires a second hand. Especially for short intermittent interactions many users therefore choose to use their bare fingers. Although convenient, this increases targeting times and error rates. We argue that the main reasons are the occlusion of the target by the user's finger and ambiguity about which part of the finger defines the selection point. We propose a pointing technique we call *Shift* that is designed to address these issues. When the user touches the screen, Shift creates a callout showing a copy of the occluded screen area and places it in a non-occluded location. The callout also shows a pointer representing the selection point of the finger. Using this visual feedback, users guide the pointer into the target by moving their finger on the screen surface and commit the target acquisition by lifting the finger. Unlike existing techniques, Shift is only invoked when necessary—over large targets no callout is created and users enjoy the full performance of an unaltered touch screen. We report the results of a user study showing that with Shift participants can select small targets with much lower error rates than an unaided touch screen and that Shift is faster than Offset Cursor for larger targets.

Author Keywords
mobile devices, touch-screens, interaction techniques, occlusion, precise target acquisition.

ACM Classification Keywords
H.5.2. User Interfaces: Input Devices and Strategies, Interaction Styles, Screen Design; D.2.2: User Interfaces

INTRODUCTION

Many pen-based devices, such as personal digital assistants (PDAs), mobile phone-PDA hybrids, and ultra mobile personal computers (UMPCs) utilize sensing technologies that can track not only a stylus, but also touch input. This makes touch input an option when pen input is not possible, such as one-handed operation [14]. Moreover, many users choose to use their finger to save the time required to retrieve the pen – especially for intermittent short interactions such as verifying a meeting time, navigating a map, or controlling a media player. However, pen-based user interfaces often contain small dense targets, making selection with a finger slow and error prone.

So what is it that users give up by not using the pen? While fingers are somewhat less accurate than pens in terms of fine control [2], accuracy is *not* the primary reason for the high error rate. In our observation, the main reason is the ambiguous selection point created by the finger's contact area *in combination with* the occlusion of the target. When selecting targets smaller than the size of the finger contact area, users start having difficulty determining whether or not they have acquired the target. Unfortunately, targets smaller than the finger's contact area are also occluded by the finger, preventing users from seeing visual feedback.

The purpose of the pen is to minimize occlusion by creating a vertical offset between the user's hand and the screen and to clearly define the selection point. Consequently, applying a technique to enhance accuracy will not solve the problem. Manipulating control display (CD) ratio [1,5] or offering in-situ zooming [1,5,17] enhance accuracy, but they do not address occlusion directly or define a clear selection point.

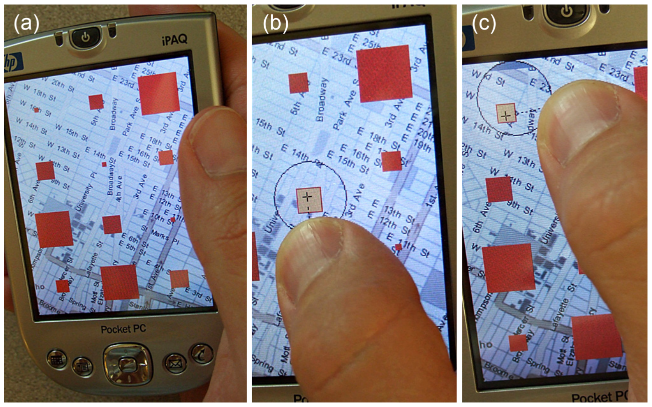


Figure 1. (a) Small targets are occluded by a user's finger. (b) The proposed *Shift* technique reveals occluded screen content in a callout displayed above the finger. This allows users to fine tune with take-off selection. (c) By adjusting the relative callout location, Shift handles targets anywhere on the screen.

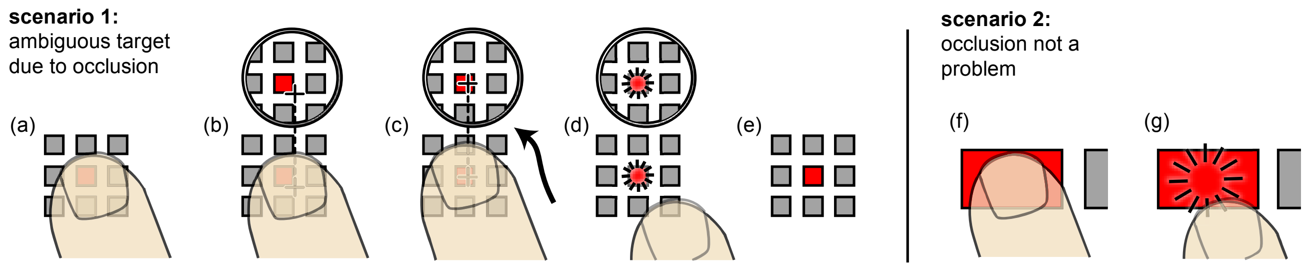


Figure 2. Shift technique walkthrough. (a-e) Scenario 1, ambiguous target selection due to occlusion: (a) on contact, Shift determines if occlusion is a problem for targets under the finger; (b) Shift responds by displaying a callout containing a copy of the occluded area with a pointer showing the finger selection point; (c) keeping the finger on the display, the user makes corrective movements until the pointer is over the target; (d) lifting the finger selects the target; and (e) removes the callout. (f-g) Scenario 2: (f) when occlusion is not a problem (g) Shift does not "escalate" and instead behaves like a regular, unmodified touch screen.

Occlusion and selection point ambiguity can be addressed with the *Offset Cursor* [18,21] (Figure 3). The Offset Cursor creates a software pointer a fixed distance above the finger's contact point. The Offset Cursor uses *take-off selection* [18,19] in which the target is selected at the point where the finger is lifted rather than where it first contacted the screen. This allows users to touch the screen anywhere and then drag the pointer into the target. Offset Cursor is in many ways a software version of a stylus: its pointer provides a unique selection point and it addresses occlusion by creating an offset between pointer and finger (in the image plane rather than above it, as the pen does).

However, the use of the Offset Cursor technique comes at a price. First, with Offset Cursor users cannot aim for the actual target anymore. Instead, they need to compensate for the offset by touching some distance away. Since there is no visual feedback until contact, users cannot always reliably predict the offset and need to iterate more. In our experimental evaluation we saw evidence of this with Offset Cursor acquisition time 1.57 times slower for targets large enough to select with the bare finger. Second, a constant offset distance and direction makes some display areas unreachable. For example, placing the pointer above the finger makes a corresponding strip along the bottom of the screen inaccessible. Although one could vary the offset direction depending on screen location, this would only exacerbate the difficulty in compensating for the offset, introducing even more corrective movement. Third, on first use, users are unlikely to expect the offset, aim directly for the actual target, and miss. While this is less of a concern in the case of a personal device, using Offset Cursor in a walk-up context like a kiosk may be questionable.

To address these disadvantages, we propose *Shift.* In addition to offsetting the pointer, Shift *offsets the screen content* to avoid all three drawbacks of Offset Cursor and leads to significantly better targeting performance.

SHIFT

Figure 2 shows a walkthrough of the Shift technique in two scenarios. Scenario 1: (a) the user touches the screen intending to acquire a small target located near other targets. Shift determines the presence of targets small enough to be occluded by the finger (see the DESIGN section for details). (b) In order to eliminate occlusion, Shift "*escalates*" by creating a *callout* that contains a copy of the occluded screen area placed in a non-occluded location on the screen. Similar to Offset Cursor, the callout includes a pointer representing the finger contact point to eliminate selection point ambiguity (c) The user fine-tunes the pointer position while maintaining contact with the screen; (d) Once the correct position is visually verified, lifting the finger causes a brief Phosphor afterglow [4] and completes the selection.

Scenario 2: (f-g) when acquiring a *large* target, Shift behaves differently. Occlusion is not a problem in this case, so Shift does not escalate by default. By lifting their finger immediately, the user makes the selection as if using an unaided touch screen.

Shift avoids the three drawbacks of Offset Cursor:

1) Shift requires interaction overhead only when really necessary, for small targets. This conditional escalation results in a significant speed-up (see the Experiment section). Conditional escalation is a property unique to Shift. If applied to Offset Cursor, users would have to perform additional movements (Figure 3a) and automatic escalation could not be determined in some cases (Figure 3b).

2) Shift does not result in any inaccessible screen areas. While the callout's default position is above the target, it can be placed elsewhere to prevent clipping by display edges (see Figure 7 and the Design section).

3) Shift behaves as touch screen users expect: it allows users to aim for the target itself. This enables walk-up scenarios. In the worst case where a user ignores the callout, Shift is no worse than a standard touch screen.

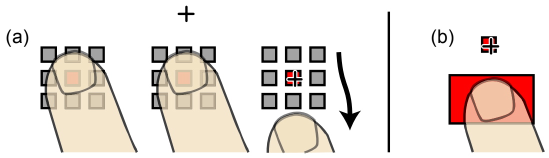


Figure 3. Shift's conditional escalation is not practical for Offset Cursor: (a) escalating to an Offset Cursor requires a large corrective movement; (b) users might avoid this by aiming below the target, but then default escalation may be ambiguous. Is the user trying to select the large target or waiting for an offset pointer to appear to help acquire the small target?

A potential drawback is that Shift requires users to visually reorient themselves at the moment of escalation. Careful design is required in order to minimize the impact of this reorientation step. Although Shift is primarily designed to address occlusion *not enhance pointing accuracy,* we show how we enhanced Shift with zooming and CD gain manipulation for high precision pointing scenarios.

RELATED WORK

Past work on improving touch screen performance and reliability can be grouped into techniques that improve the precision of finger pointing and those that avoid occlusion.

Specializing the User Interface for Fingers

Maximum usability can be achieved with user interfaces designed directly for the capabilities of the finger [20]. Such user interfaces typically avoid small targets in the first place. Karlson et al. explore this strategy with AppLens and LaunchTile [13], two thumb-specialized designs for PDA application shells. However, creating input-specialized versions of applications is expensive and will often be difficult to do for legacy applications. Also, if both pen and finger input are used intermittently, then ideally two sets of input-specialized interfaces would need to be available.

Increasing Finger Precision

A large corpus of work has focused on techniques for precise touch screen interaction, some of which also address the issue of finger occlusion indirectly.

Albinsson and Zhai [1] use widgets to fine-tune a finger's selection point by zooming, reducing CD gain, and discrete pixel-by-pixel adjustment. Compared to an un-stabilized Offset Cursor, their techniques introduce fewer errors for targets smaller than 0.8 mm with zooming performing the fastest. However, their techniques are slower than Offset Cursor for targets larger than 3.2 mm, likely because multiple taps are required even when no adjustment is necessary. Occlusion was not explicitly considered in their techniques, but some of them can help because they use widgets placed away from the target. Albinsson and Zhai discuss the problem of occlusion when using a technique with a button mounted above the selection point. Olwal and Feiner [17] describe another precise touch screen technique invoked using a rubbing motion, which produces a zoomed fisheye view to expand the targets. No evaluation was conducted.

Benko, et al. [5] explore pixel-accurate selection techniques resilient to input noise on a vision-based multi-touch screen device. All techniques default to using direct touch using their SimPress click technique. An optional second touch point invokes and controls the precision techniques. They compared two CD-gain manipulation techniques, zooming, and a two-fingered Offset Cursor. They found zooming more precise for 1 pixel targets, with the two-fingered offset less precise for target sizes up to 8 pixels. The two-fingered Offset Cursor had comparable task times to zoom. Their style of multiple finger contact would not be appropriate for small, hand-held devices.

Researchers have also proposed simplifying target acquisition by snapping the pointer to nearby targets (*Bubble Cursor* [10]), increasing the size of targets in motor space (*Semantic Pointing* [6]), or making the pointer select targets within a fixed area (*Prince Technique* [12]). These techniques reach optimum performance for loosely spaced targets, a situation not common on small device UIs. They are also inappropriate when pointing in continuous spaces like a map or text editor where targets are not well defined.

Avoiding Finger Occlusion

As discussed earlier, Potter et al.'s Offset Cursor is designed explicitly to eliminate finger occlusion [18]. In an evaluation using a single target size of 6.35mm, Offset Cursor with take-off selection had lower error rates compared to simple direct touch with first-contact or land-on selection. However, it was found to be significantly slower than the first-contact selection strategy. Since their experimental design does not separate effects of pointer offset and selection strategy, it is difficult to tell if this is due to the offset. In a pen-based study, Ren and Moriya [19] confirmed take-off without an offset to be advantageous in terms of error. Other researchers have extended Offset Cursor for multiple touch points to control offset distance and direction [5,8].

Sears and Shneiderman [21] add stabilization to the Offset Cursor for more precise pointing. They use three discrete CD-gain levels (0, <1, 1) set according to the distance from the initial touch point. Experimental results showed comparable times and errors for stabilized and un-stabilized conditions with the exception of 1 pixel targets.

A variation of the Offset Cursor is a "touch pointer" which provides a visible, finger-sized handle to manipulate the offset pointer. LaunchTile [13] includes this type of pointer for selecting text and map locations (scenarios where making all targets finger sized was not practical). Other examples of touch pointers are the "Touch Mouse" used in Windows Vista™ and Albinson and Zhai's precision handle [1]. Although a touch pointer makes the offset visible, it also makes pointing a compound task: first acquire the handle, drag it to the desired location, then fine-tune and tap to make the actual selection. Another issue is that it occupies permanent space on the display, making the touch pointer less suitable for small screen devices.

Some applications which create multiple views of the same content for navigation or accessibility could be repurposed to address occlusion. For example, the Microsoft Windows™ Magnifier Tool creates an additional view of the area around the pointer. However, this permanently occupies screen space at a fixed position, unlike Shift which uses conditional escalation to place a second view near the finger.

Recently, researchers have explored using the underside of an interactive table to address occlusion [23]. However, this approach requires specialized hardware and usability with a hand-held device was not evaluated.

DESIGN

In order to guide the design process, we created a model of Shift's expected targeting performance (Figure 4). This model is the basis for all our hypotheses and it guided us through several rounds of pilot studies.

We formulated our hypotheses within the context of Offset Cursor and unaided touch screen selection. The simplicity of unaided touch screen input make it fast across all target sizes, but there is an approximate threshold size where occlusion makes selecting smaller targets error prone [19]. We call this the *occlusion threshold*. Offset Cursor avoids these problems by offering a defined selection point and avoiding occlusion. In exchange, however, users spend additional time estimating the offset distance and fine-tuning their selection position. During pilot testing we were surprised to observe that the time loss was not limited to small targets where occlusion was a problem, but affected all target sizes we tested including targets as large as 41mm. We discuss this in detail in the experiment section.

We hypothesized that Shift performance should differ depending on target size. For targets smaller than the occlusion threshold, Shift should perform roughly the same as Offset Cursor since both offer improved accuracy at the expense of additional user effort. However, we did not know how long Shift's visual reorientation step would take in comparison to Offset Cursor's distance estimation step. For large targets, however, we expected a clear performance benefit for Shift over Offset Cursor since without escalation Shift works like an unmodified touch screen.

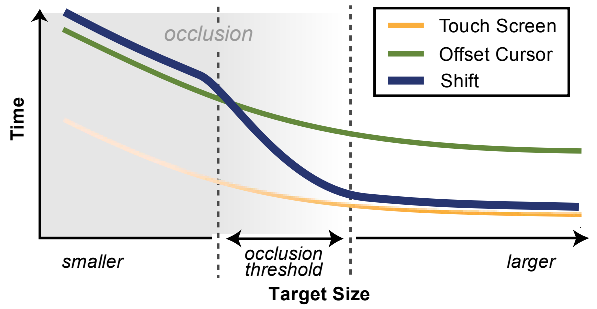


Figure 4. Expected targeting performance model. Shift should behave similar to Offset Cursor for small targets and identical to unaided touch screen with large targets. Touch screen is fast, but high errors with small targets make it impractical.

Shifting too much screen space can disorient users

Shift works by displaying a copy of the occluded area in a non-occluded location. There are many ways to do this, and we needed to determine aspects like the size of the copied area and where it should be placed. With our first design, we tried to provide as much context around the target as possible and to minimize clutter. We created a simple design which on finger contact translated the entire display up by 22mm (Figure 5).

To verify this design, we conducted a pilot study comparing it with *Offset Cursor* and unaided touch screen interaction. While results showed that the targeting performance of this particular Shift technique was comparable to Offset Cursor for small targets, Shifts targeting performance did not hug the touch screen curve for large targets as we had hypothesized. Instead it stayed close to Offset Cursor throughout. The post study questionnaire revealed that participants found the movement of the entire screen distracting, likely affecting their performance. One participant commented "I felt that the targets were jumping away from my finger."

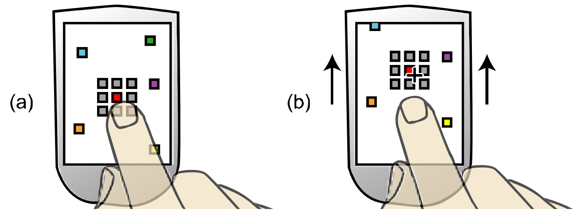


Figure 5. Initial design for Shift: (a) on finger contact; (b) the entire display is translated upwards by a fixed distance.

Based on these findings we redesigned Shift's visuals by replacing the motion of the entire display with a callout. Figure 6 shows several designs we considered. We selected (d) as the final design which uses a circular frame, 26mm in diameter, to reference the shape of the occluded area under the finger. The round shape preserves the target's immediate proximity, while minimizing overall surface. The round shape also made the callout stand out among typically rectangular screen layouts, making additional highlighting (b and c) dispensable.

The final design connects the shifted pointer and actual contact point under the finger with a dashed line. Despite its simplicity, this seemed sufficient for communicating the origin of the callout, allowing us to drop visually heavier styles, such as a cartoon bubble callout (b and c) or a rubber band (a) à la drag-and-pop [3].

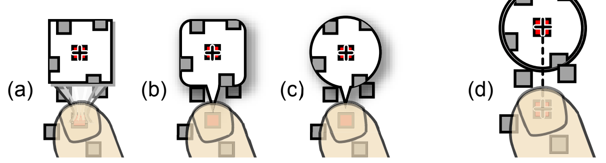


Figure 6. (a-c) Three prototype designs for callout visuals to reduce "jump away" effect and (d) the final design.

Callout placement to handle screen edges

Our design goals for callout placement were to minimize occlusion as well as to maximize predictability in order to accelerate visual re-orientation. To minimize eye travel, we place the callout near the finger. As the default position, we placed the callout 22mm above the initial touch point. This corresponds to the strategy followed by Offset Cursor and covers most cases minimizing occlusion with the hand and finger for most hand postures (Figure 7a). Along left and right edges clipping is avoided by positioning the callout further towards the middle of the screen (Figure 7b). If touch occurs near the top edge of the display we avoid clipping by positioning the callout to the left and if that is not possible to the right (Figure 7c,d). This can be reversed for left-handed users (using handedness detection [11]).

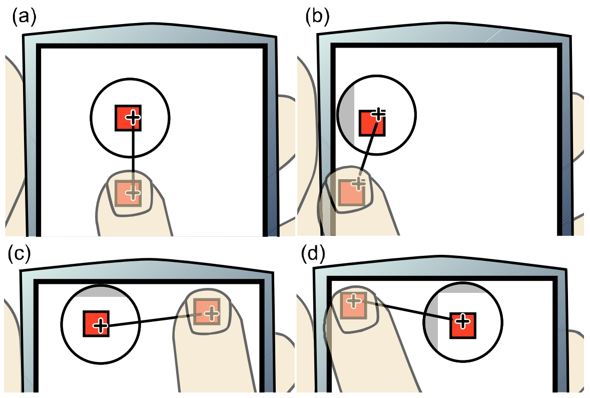


Figure 7. Callout placement: (a) default position; (b) avoiding clipping at edges; (c and d) avoiding clipping at top.

Escalation based on hesitation and selection ambiguity

Shift cannot always know whether the user needs support in an upcoming targeting attempt. By using dwell time, the ultimate decision about whether or not to escalate is left to the user. In the complete absence of additional knowledge about target size and locations, a fixed timeout is used such as 300ms. But if tighter integration makes target sizes and locations available, Shift can determine dwell time based on *selection ambiguity*. Mankoff et al. also discuss the problem of target selection ambiguity. In their system, if the first click is ambiguous, a "magnifier" is displayed and a second click is required to resolve the ambiguity [15].

We calculate selection ambiguity by comparing the smallest target size found under the finger with the occlusion threshold size (Figure 8). When the target is small compared to the occlusion threshold, the selection ambiguity is high. In this case, we can set the dwell timeout to be very short and escalate immediately. However, if the target is much larger than the occlusion threshold, then occlusion is not a problem and escalation is not necessary. The timeout can then be set to a longer time enabling users to take advantage of simple, direct touch. For targets around the same size as the occlusion threshold, the degree of selection ambiguity is itself ambiguous (the user may or may not need escalation depending on their confidence in their selection). In this case, the dwell timeout occurs after a short delay just long enough to control escalation invocation with hesitation. If they want to escalate, they hesitate by holding down for a moment. To avoid escalation, they lift up immediately.

Our implementation uses the difference between S_T, the smallest dimension of the smallest target under the finger, and S_F , the occlusion threshold size. S_T - S_F is mapped to a dwell time using a logistic function with parameters of a=1, m=0, n=4, and τ=3 (Figure 8). This produces a smooth curve mapping small targets to ~0ms, large to ~1200ms and targets near the occlusion threshold to about 300ms.

Estimating Occlusion Threshold

The occlusion threshold is roughly related to the finger contact area, but touch sensitive screens commonly used on PDAs and UMPCs only report a single input point and not the finger contact area. So we form an estimate of the occlusion threshold S_F over time, based on the target sizes for which they do and do not use escalation. We begin with an initial guess S_F, then increase S_F by s if the user escalates when $S_F < S_T$ and decrease S_F by s if the user does not escalate and $S_F > S_T$. We define $s = w|S_F - S_T|$, where w is a hand tuned weight to smooth the estimate over time. We found that w=0.125 gave a good balance between smoothness and learning rate.

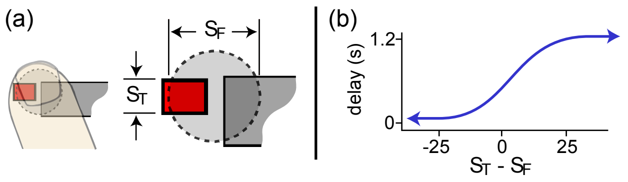


Figure 8. Ambiguity estimation for escalation: (a) the occlusion threshold diameter S_F and the smallest dimension of the smallest target found under the finger S_T ; (b) logistic function maps the difference S_T - S_F to a dwell timeout.

A potential benefit of this scheme is that if the user prefers to use their fingernail, S_F will shrink so that escalation is instant only for very small targets. For devices that can sense if the stylus is in the device holster, our approach allows learning independent S_F values for finger and pen input, respectively. In the absence of this sensor data, setting w to a high value allows learning a new S_F quickly to respond to changes in the user's input style.

Correcting for User's Perceived Input Point

The single selection point computed by a resistive touch screen is placed roughly at the mean finger contact area [21] (Figure 9b). Benko et al. suggest that many users perceive the selection point of their finger as being located near the top of the finger tip [5] (Figure 9a). In an examination of log data from our pilot, we found that contact points were often slightly below the intended target. Since Shift's escalated pointer position is displayed relative to the initial contact point, we adapt the location to reflect the user's perceived contact point.

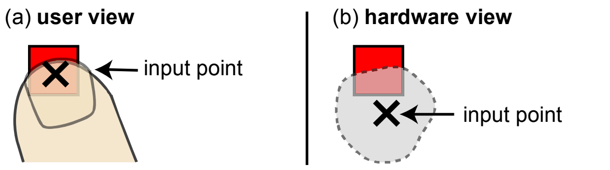


Figure 9. Perceived input point: (a) users expect the input point to be near the tip of their finger; (b) hardware places the input at the centre of the finger contact area.

The vision-based touch screen used by Benko et al. [5] allowed them to place the pointer according to the finger's actual contact area. Shift works with more common resistive touch screens by adjusting the input position based on a single contact point. We continuously refine an estimate of a correction vector V mapping the hardware input point to the user's perceived input point. We update V by adding a weighted vector between the corrected final lift-off point P_2 and initial contact point P_1: $V_{t+1} = V_t + w(P_2 - P_1)$, where w

is a hand-tuned weight. We found that *w=0.33* was able to smooth the estimate without making the iterative refinement too slow. Note that this type of adaptation is not possible with the conventional Offset Cursor since it is absolutely positioned.

In informal user evaluations, we found that this reduced fine-tuning time after the estimate of *V* converges, allowing users to simply verify the selected target without further adjustment. But unlike the finger, the contact shape of the thumb tends to change depending on the contact location on the display. This makes a single adjustment vector insufficient. A linear interpolation between location specific adjustment vectors may alleviate this problem.

Pointer stabilization

We found that the hardware designed for pen input can be noisy when operated with a finger (although recently researchers claim pointer stabilization is unnecessary with modern touch screens [1]). For our reference implementation we addressed this by adding a dynamic recursive low pass filter [22] which increases pointer stability during slow corrective movements without introducing any significant lag during high velocity movements. It works by interpolating between two low-pass filters according to estimated velocity – we found that cut-off frequencies of 5 and 18 Hz interpolated between 150 and 560 mm/s produced good results. Unlike Sears and Shneiderman's stabilization technique [21], our filter eliminates noise without artificially changing finger accuracy by CD-gain manipulation. We discuss further targeting enhancement techniques for Shift, such as CD ratio adjustment and zoom, in a later section.

PROTOTYPE IMPLEMENTATION

Our Shift prototype runs on the Windows Mobile Platform and is implemented in C# using the .NET Compact Framework 2.0. Our primary platform is an IPAQ 4100 with a 400 MHz processor. With standard GDI+ graphics our prototype application runs at 15 FPS. We also adapted our prototype application to run on a Windows Smart phone with a touch sensitive display and a UMPC.

EXPERIMENTAL EVALUATION

In this experiment, participants acquired targets of different sizes and positions using *Shift*, *Offset Cursor*, and unaided touch pointing (*Touch*). Based on our model of expected targeting performance (Figure 4), we hypothesized that: (1) Shift and Offset Cursor would outperform Touch in terms of error rate with smaller targets; and (2) that Shift would outperform Offset Cursor for larger targets in terms of task time (at comparable error rates).

Task and Stimuli

Participants were presented with a series of individual target selection trials. Six different target sizes were used, with each target positioned a constant distance away at four different angles. Participants were instructed to acquire these targets as quickly and accurately as possible.

Participants acquired targets with the index finger of their dominant hand while holding the device in their non-dominant hand (Figure 10). An earlier pilot study had found similar patterns of performance between one-handed thumb targeting and two-handed finger use (except that the thumb condition showed a higher variance due to thumb ergonomic issues [14]). We used separate conditions for *FingerTip* and *Fingernail* because our pilot studies had found that different participants preferred using their finger tip while others preferred fingernail.

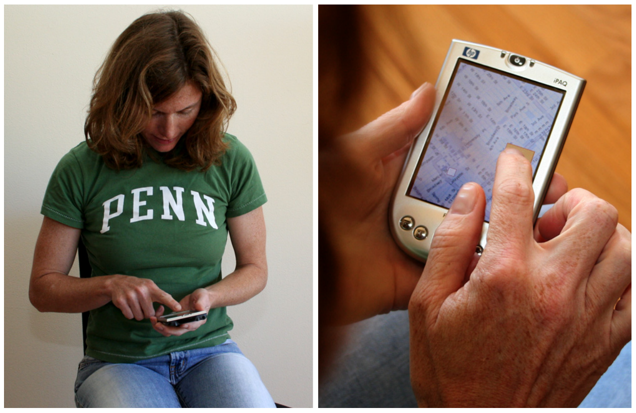


Figure 10. Apparatus: participants acquired targets on an IPAQ 4100 PDA using their index finger.

At the beginning of each trial, a solid red 48px (48 pixel by 48 pixel) *start button* was displayed along with the target, which was rendered as white with a red border (Figure 11a). The target was placed diagonally, 120px away from the start button. Targets were never placed at screen edges so all targets were reachable using Offset Cursor. Stimuli were displayed in front of a street map background for increased ecological validity. Participants selected the start button using the currently active technique condition. Once selected, the start button disappeared and the target turned solid red (Figure 11b). When the pointer was over the target, the target provided visual feedback by turning yellow (Figure 11c). The trial was completed when participants lifted their finger. Successful target acquisition was confirmed with a click sound; unsuccessful attempts resulted in an error sound. Since we expected the *Touch* condition to have high error rates with small targets, participants advanced to the next trial regardless of error. We recorded task times, errors, and all movement and escalation events.

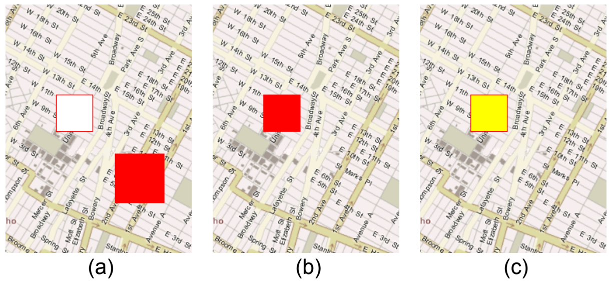


Figure 11. Experimental task stimuli: (a) red start button and white target with red border; (b) start button disappears when selected, target turns red; (c) visual feedback when *over* target.

Design

A repeated measures within-participant factorial design was used. The independent variables were *Technique* (*Shift*, *Offset*, and *Touch*), *Contact* method (*FingerTip* and *Fingernail*), target *Direction* (*NW*, *NE*, *SW*, *SE*), and target *Size* (6, 12, 18, 24, 48, 96 px) (measuring 2.6 to 41.9 mm on screen). Piloting had indicated that many participants chose to escalate for 12px, but not for 24px targets. The 18px target was inserted into the otherwise geometric row of target sizes to help pinpoint the exact location of this "crossover." We did not see any targets smaller than 12px when analyzing existing pen applications; and even with a pen, targets this small tend to be difficult to acquire [19]. Nonetheless we included the additional 6px target size to learn more about the theoretical limits of our techniques.

Presentation of *Technique* and *Contact* was counterbalanced across participants. The 6 *Target Sizes* were paired with each of the 4 *Directions* and presented in random order within each block. The experiment had 1 practice block and 3 timed blocks for each *Technique* and *Contact* combination.

In summary, the experimental design was:

3 *Techniques* (*Shift*, *Offset*, and *Touch)* ×
2 *Contact* styles (*FingerTip* and *Fingernail)* ×
3 *Blocks* ×
6 *Sizes* (6, 12, 18, 24, 48, 96 px) ×
4 *Directions* (*NW*, *NE*, *SW*, *SE*)
= 432 data points per participant

Apparatus

The experiment was conducted on the IPAQ PDA described above. It has a 550 × 730mm, 240 × 320px, display producing an effective resolution of 0.436px/mm.

The Offset Cursor and Shift techniques functioned as described earlier, except Shift's perceived input point adaptation was enabled only during training, to keep timed trials consistent. Escalation on target ambiguity operated as described earlier, but with the occlusion threshold size fixed at 24px. This made dwell times of 0, 5, 39, 240, 1198, and 1199 ms for the 6px through 96px targets respectively.

Participants

Twelve volunteers (3 female), ranging in age from 24 to 41 years, were recruited from our institution. Each received a lunch coupon for our cafeteria as a gratuity for their time. Two participants were left handed. All participants had some experience with pen-based PDAs.

Hypothesis

Based on our model of expected targeting performance (Figure 4), we had the following two hypotheses:

(H1) For small targets we expected *Shift* and *Offset Cursor* to outperform *Touch* in terms of error rate, but at the expense of increased task time for fine tuning.

(H2) For large targets, we expected *Shift* to outperform *Offset* in terms of task time at comparable error rates. The reason is that we expected the screen offset to require cognitive effort, while *Shift's* optional escalation would avoid any overhead for large targets.

Based on our observations during pilot testing, we speculated that for small targets Shift's visual re-orientation might take longer than Offset Cursor's correction of the unknown touch location, but we had no hypothesis about the extent of this difference. Given that *Shift* without escalation is identical to *Touch*, we did not expect to see any performance differences between these two interface conditions for large targets.

Results

Repeated measures analysis of variance showed that the order of presentation of the three *Techniques* and two *Contact* styles had no significant effect on selection time or error rate, indicating that a within-subjects design was appropriate. A 3 × 2 × 3 (*Technique* × *Contact* × *Block*) within subjects analysis of variance found a significant main effect for *Block* on selection time ($F_{2,18} = 6.393$, $p < .01$) indicating the presence of a learning effect. Post hoc analysis revealed that *Block* 1 was significantly slower than *Blocks* 2 and 3, so *Block* 1 was not included in subsequent analysis.

Error Rate

We aggregated selection errors to perform a 3 × 2 × 6 (*Technique* × *Contact* × *Size*) within subjects analysis of variance. There were significant main effects for *Size* ($F_{5,45} = 58.757$, $p < .001$) and *Technique* ($F_{2,18} = 58.757$, $p < .001$). Somewhat surprisingly, there was no significant main effect for *Contact*—one might have expected the Fingernail condition to have a lower error rate. However, there was a *Technique* × *Contact* × *Size* interaction ($F_{10,90} = 5.710$, $p < .01$) illustrated in Figure 12. Post hoc multiple means comparison tests showed that *Touch* had significantly higher error rates compared to *Offset* and *Shift* for target *Sizes* 6 and 12px in both *Contact* conditions (all $p < .01$ except for *Shift* with *Fingernail*, $p < .05$). This supports our first hypothesis. With error rates as high as 81% (*FingerTip*) and 63% (*Fingernail*) for the smallest targets, this confirms the results of earlier work [18] showing that high error rates make *Touch* unreliable for small targets.

No significant differences in error rate were found between *Offset* and *Shift*. Error rates for *Offset* and *Shift* at 6 and 12px targets were somewhat high (between 4 and 14%). Although error rates in that range are not uncommon for small targets even with a pen [19], part of the error rate might be explained by noisy lift-off. We observed some participants missing selections because lifting the finger caused a brief rolling motion, sometimes moving the pointer away from a correctly acquired target. Future versions of Offset Cursor and Shift should correct for this bias by discarding motion immediately before take-off [7].

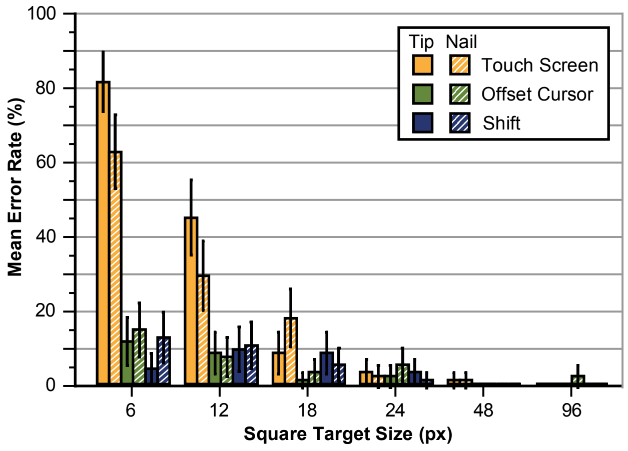


Figure 12. Mean Error Rate for *Technique* and *Contact*. Error bars represent 95% confidence interval.

Selection Time

Task time was measured from the moment the finger was lifted off the start button to the moment the finger was lifted from the target. Trials with selection errors were excluded from the selection time analysis. To correct for skewing common to human response time data, we used the median response of aggregated repetitions for each participant. To verify that we could aggregate across *Directions*, we performed a 3 × 2 × 4 (*Technique* × *Contact* × *Direction*) within subjects analysis of variance and found no significant main effects or interactions. Due to the very high error rates in the *Size* 6px condition with *Touch*, the main analysis included only the 5 larger target *Sizes*.

We performed a 3 × 2 × 5 (*Technique* × *Contact* × *Size*) within subjects analysis of variance on median response times aggregated across *Blocks* and *Directions*. Significant main effects were found for *Technique* ($F_{2,18} = 39.006$, $p < .001$) and *Size* ($F_{4,36} = 125.492$, $p < .001$). However, most relevant is the significant *Technique* × *Contact* × *Size* interaction ($F_{2.48,72} = 5.027$, $p = .011$). Post hoc multiple means comparison tests confirmed that *Shift* is faster than *Offset* for large targets. The speed crossover point depended on the *Contact* condition. For *FingerTip*, *Offset* and *Shift* were significantly slower than *Touch* at *Size*s 12 to 24px (all $p < 0.01$), but *Offset* was slower than both *Shift* and *Touch* at Sizes above 24px (all $p < 0.001$). For *Fingernail*, *Offset* and *Shift* were significantly slower than *Touch* at *Size* 12px (all $p < 0.02$), but *Offset* was slower than *Shift* and *Touch* for Sizes above 12px (all $p < 0.03$). This supports hypothesis H2. Furthermore, as Figure 13 suggests, task times with Shift were indeed comparable to *Touch* for larger targets and comparable to *Offset* for smaller targets.

In summary, in the *Fingernail* condition the mean trial times of *Shift* (and *Touch*) for targets 18px and larger were 700ms (and 628ms) compared to *Offset* with a mean time of 1027ms—a 32% improvement for *Shift*. The *FingerTip* condition showed the same trend, but with the crossover occurring at 48px with mean times for *Shift* (and *Touch*) of 609ms (and 578ms) compared to *Offset* with 849ms – a 28% improvement for *Shift*.

The high *Size* 6px target error rates with *Touch* prevented us from including it in the main comparison, so we performed a 2 × 2 (*Technique* × *Contact*) within subjects analysis of variance for *Shift* and *Offset* for the *Size* 6px target only. Unlike the other small target sizes above 6px, we found a significant main effect for *Technique* ($F_{1,9} = 12.76$, $p < .01$) with *Offset* 316ms faster than *Shift*. No interactions were found. This suggests that re-orientation during *Shift's* escalation does require more time than distance adjustment with *Offset Cursor* for very small targets. The practical implications of this difference should be small though. As mentioned earlier, targets below 12px in existing mobile applications are difficult to acquire with a pen and are relatively rare.

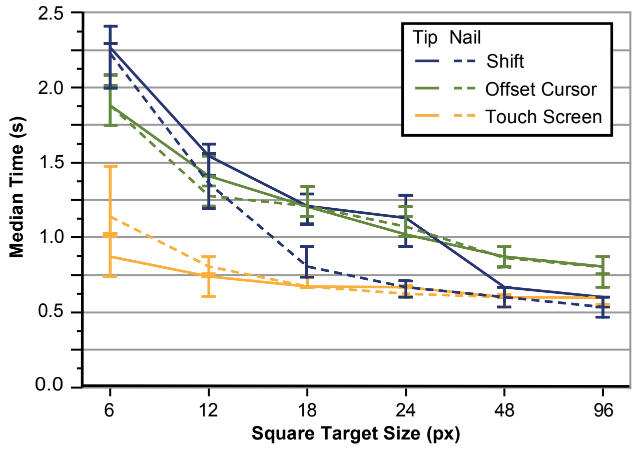


Figure 13. Median Selection Time for *Technique* and *Contact*. Error bars represent 95% confidence interval.

Selection times with Shift can be further understood by looking at escalation usage (Figure 14). For very small targets, *Shift* escalation was used in most trials resulting in performance similar to *Offset*. For large targets, escalation was rarely used making performance identical to Touch. For midsized targets near the occlusion threshold, participants used a mix of escalation and non-escalation.

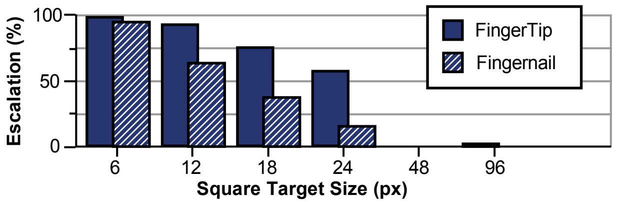


Figure 14. Percentage of targeting attempts in which participants chose to escalate with Shift

Subjective Preference

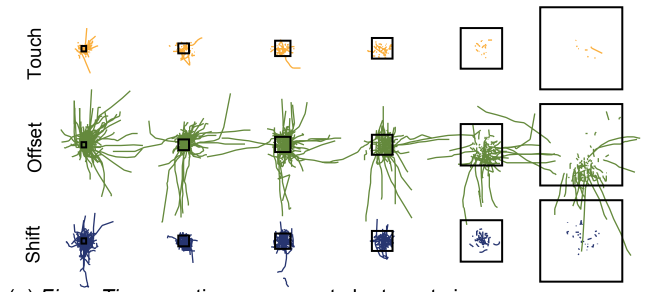
In the post-study questionnaire participants ranked the three *Techniques* by preference. For use with *FingerTip*, 7 participants preferred *Shift*, 5 *Offset*, and 0 *Touch*. For *Fingernail*, 7 participants preferred *Shift*, 3 *Offset*, and 2 *Touch*. In addition, we asked participants about their preference for *FingerTip* vs. *Fingernail;* 5 participants commented that although they felt *Fingernail* was more accurate, it seemed unnatural and uncomfortable compared to using *FingerTip*.

Discussion

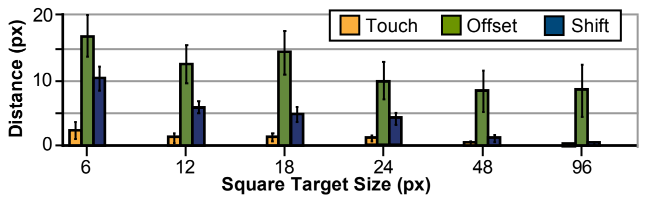
Our experimental results support our hypothesis, which we based on our theoretical performance model. Like Offset Cursor, Shift's occlusion avoidance benefit enables reliable selection of small targets. But unlike Offset Cursor, Shift's hybrid nature preserves the speed and simplicity of direct touch interaction when occlusion is not a problem.

Our results also confirmed our earlier observations about Offset Cursor, namely that it impacts task times even for large targets. For targets 24px and greater *Offset* was 1.57 times slower than *Touch* with mean median times of 938ms and 597ms respectively (Figure 13). This is somewhat surprising since estimating the offset distance should be much easier with large targets, given the increased error tolerance.

One possible reason why Offset Cursor is slower may be because users often overshoot or undershoot the target, resulting in a higher *net correction distance* – the screen distance between initial contact and final lift-off (Figure 15b). For larger targets, Offset Cursor still has a high net correction distance even though one might expect users to acquire large targets without correction. This suggests that users may have difficulty estimating the offset distance. Another supporting observation is how users tend to tap near the bottom targets (Figure 15a). With *Shift* and *Touch*, we can see the contact pattern is more centered since no pre-contact distance estimation is needed.



(a) *FingerTip* corrective movements by target size



(b) *FingerTip* net correction distance by target size

Figure 15. *FingerTip* corrective movements for target *Sizes* and *Technique*. *Fingernail* results are similar.

HIGH ACCURACY SELECTION ENHANCEMENTS

As stated earlier, the purpose of Shift is to enable users to acquire targets by avoiding target occlusion, not necessarily enhancing targeting *precision*. Our study shows that the basic version of Shift described above allows for the acquisition of targets as small as 6 pixels (2.6 mm). Some situations, however, may require users to acquire targets smaller than that. Fortunately, Shift lends itself well to precision enhancements like zooming and control display (CD) ratio manipulation.

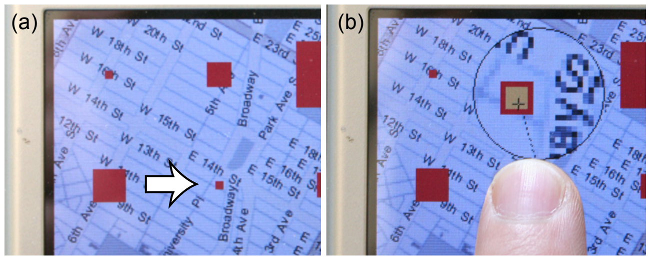


Figure 16. Shift with zooming enhancement: (a) before finger contact; (b) after escalation with 4× magnification in callout.

Zooming

Our enhanced high-precision version of Shift is visually identical to regular Shift, except that the callout is magnified (Figure 16). The higher the callout's magnification, the less content the callout will show. To at least partially compensate for this, we increase the callout diameter from 26mm to 44mm. In order to allow users to reach content beyond that, we changed the callout so that it now travels with the finger, similar to a tracking menu [9].

Since the finger is no longer mapped directly with pointer position, the callout itself must be moved so that it does not become occluded during the corrective phase. As before, the initial position of the callout is placed relative to the initial contact point. If the contact point moves beyond a threshold diameter, the callout moves along with the finger similar to a tracking menu. This allows fine-tuning beyond the initial area covered by the frame if the initial contact point was too far from the desired target given the increased zoom space (or increased motor space with high CD ratios).

Because of its space demands, we found the use of zooming on a small screen device to be limited to magnification factors of up to 4×. While such a magnification will assure the visibility of a pixel-sized target, it may not be enough to allow for reliable target acquisition. We therefore complemented zooming with an enhancement in CD ratio.

CD ratio enhancement

Touch screens are typically operated with a CD ratio of 1 in which the pointer position is mapped 1:1 with the finger input position. Especially for systems that do not support a tracking state, a 1:1 mapping is important because it allows users to aim for a target. However, once the user's finger is in contact with the screen, a pointer can be displayed providing users with visual feedback. Then, finger movement can control the pointer in a relative manner, with the pointer moving faster or slower than the finger directing it.

Shift's high precision version offers CD ratios of up to 8 when escalated. This means that pointer movement across the screen is slowed down expanding a 1px target to 8px in motor space. Shift's CD ratio is changed on escalation which is somewhat similar to Benko et al. [5] except that they require a second finger to select from multiple levels of CD ratio. Other researchers have adjusted CD ratio with a pantograph-like handle [1] or based on distance from the initial touch point for the purpose of stabilization [21].

As explained in the previous section, we position the callout to avoid occlusion with the finger. This is done regardless of the target's original position. In some cases moving the finger makes the original target position no longer occluded. However, in a pilot user study, participants told us they always used the callout for visual feedback, even if they realized that the target in the main display was visible.

Since the touch sensitive display has a finite input area, increasing CD ratio above 1 reduces the *range* of motor space to 1/CD of display space. This can be a problem if the initial contact point is X pixels away from the edge of the display and more than X/CD pixels further away from the target. Since selection is on lift-off, there is no way for the user to select the target. Possible solutions would be to snap to a point closer to the edge where all intermediate pixels were selectable or using pointer acceleration so that a quick succession of long slow and short fast movements could simulate clutching. However, in practice, we did not find this to be a significant problem when selecting targets away from the edges.

CONCLUSION

Shift enables the operation of a pen-based device, such as a PDA or UMPC, with fingers. Our experimental results show that Shift's conditional escalation overcomes occlusion problems and allows users to select small targets reliably. Unlike Offset Cursor [18], Shift preserves the speed and simplicity of direct touch interaction for large targets.

While the user study reported in this paper focused on these quantitative differences, another benefit mentioned earlier might have an even bigger impact on Shift's deployment: By allowing users to aim for the actual target, Shift remains compatible with regular pen and touch input. This compatibility keeps the interaction consistent when switching back and forth between pen and touch. And, maybe most importantly, it makes it easy to deploy Shift in walk-up scenarios or to retrofit existing systems. We plan to study such scenarios as our future work.

ACKNOWLEDGEMENTS

We thank members of the ASI and VIBE groups at Microsoft Research with special thanks to Raman Sarin for the experimental framework and Ed Cutrell for statistical analysis help. We also thank David Thiel for producing the video, and all of our study participants.

REFERENCES

1. Albinsson, P. Zhai, S. (2003). High precision touch screen interaction. In *Proc. of CHI' 2003*, 105-112.
2. Balakrishnan, R. MacKenzie, S. (1997). Performance differences in the fingers, wrist, and forearm in computer input control. In *Proc. of CHI'97*, 303-310.
3. Baudisch, P., Cutrell, E., Robbins, D., Czerwinski, M., Tandler, P. Bederson, B., Zierlinger, A. (2003). Drag-and-Pop and Drag-and-Pick: Techniques for Accessing Remote Screen Content on Touch and Pen-operated Systems. In *Proc. of Interact'03*, 57-64.
4. Baudisch, P., Tan, D., Collomb, M., Robbins, D., Hinckley, K., Agrawala, M., Zhao, S., and Ramos, G. (2006). Phosphor: Explaining Transitions in the User Interface Using Afterglow Effects. In *Proc. of UIST'06*, 169-178.
5. Benko, H., Wilson, A., Baudisch, P. (2006). Precise selection techniques for multi-touch screens. *Proc. of CHI' 06*, 1263-1272.
6. Blanch, R. Guiard, Y., Beaudouin-Lafon, M. (2004). Semantic Pointing: Improving Target Acquisition with ControlDisplay Ratio Adaptation. In *Proc. of. CHI'04*, 519-526.
7. Buxton, W., Hill, R., Rowley, P. (1985). Issues and Techniques in Touch-Sensitive Tablet Input. *Computer Graphics*, 19(3):215-224.
8. Esenther, A., Ryall, K. (2006). Fluid DTMouse: better mouse support for touch-based interactions. *Proc. AVI '06*, 112-115.
9. Fitzmaurice, G., Khan, A., Pieké, R., Buxton, B., and Kurtenbach, G. (2003). Tracking menus. In *Proc. of UIST '03*, 71-79.
10. Grossman, T, Balakrishnan, R. (2005). The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. In *Proc. of CHI'05*, 281-290.
11. Hancock, M., Booth, K. (2004). Improving Menu Placement Strategies for Pen Input. In *Proc. of GI'04*, 221-230.
12. Kabbash, P., Buxton, W. (1995). The "Prince" technique: Fitts' law and selection using area cursor. In *Proc. of CHI'95*, 273-279.
13. Karlson, K., Bederson, B., SanGiovanni, J. (2005). AppLens and LaunchTile: two designs for one-handed thumb use on small devices. In *Proc. of CHI'05*, 201-210.
14. Karlson, A., Bederson, B., Contreras-Vidal, J. (2006) Understanding Single-Handed Mobile Device Interaction. *Tech Report HCIL-2006-02*.
15. Mankoff J., Hudson S., Abowd G. (2000). Interaction techniques for ambiguity resolution in recognition-based interfaces. In *Proc. of UIST '00*, 11-20.
16. Matsushita, N., Ayatsuka, Y., Rekimoto, J. (2000). Dual touch: A two-handed interface for pen-based PDAs. In *Proc. of UIST'00*, 211-212.
17. Olwal, A., Feiner S. (2003) Rubbing the Fisheye: precise touch-screen interaction with gestures and fisheye views. In *Conf. Supp. of UIST'03*, 83-84.
18. Potter, R., Weldon, L., Shneiderman, B. (1988). Improving the accuracy of touch screens: an experimental evaluation of three strategies. *Proc. CHI'* 88, 27-32.
19. Ren, X., Moriya, S. (2000). Improving selection performance on pen-based systems: a study of pen-based interaction for selection tasks. *ACM TOCHI.* 7(3):384-416.
20. Shneiderman, B. (1991). Touch Screens Now Offer Compelling Uses. *IEEE Softw*. 8, 2. 93-94, 107.
21. Sears, A., Shneiderman, B. (1991). High precision touchscreens: design strategies and comparisons with a mouse. *Int. J. Man-Mach. Stud.* 34(4):593-613.
22. Vogel, D. Balakrishnan, R. (2005). Distant freehand pointing and clicking on very large, high resolution displays. In *Proc. of UIST '05*, 33-42.
23. Wigdor, D., Leigh, D., Forlines, C., Shipman, S., Barnwell, J., Balakrishnan, R., Shen, C. (2006). Under the Table Interaction. In *Proc. of UIST'06*, 259-268.