Olwal, A., Feiner, S., and Heyman, S. Rubbing and Tapping for Precise and Rapid Selection on Touch-Screen Displays.
Proceedings of *CHI 2008 (SIGCHI Conference on Human Factors in Computing Systems)*, Florence, Italy, April 5-10, 2008, pp. 295-304.

Rubbing and Tapping for Precise and Rapid Selection on Touch-Screen Displays

Alex Olwal[1] **Steven Feiner**[2] **Susanna Heyman**[1]

[1]School of Computer Science and Communication
KTH (Royal Institute of Technology), Stockholm

[2]Department of Computer Science
Columbia University, New York, NY

alx@csc.kth.se, feiner@cs.columbia.edu, susanna.heyman@gmail.com

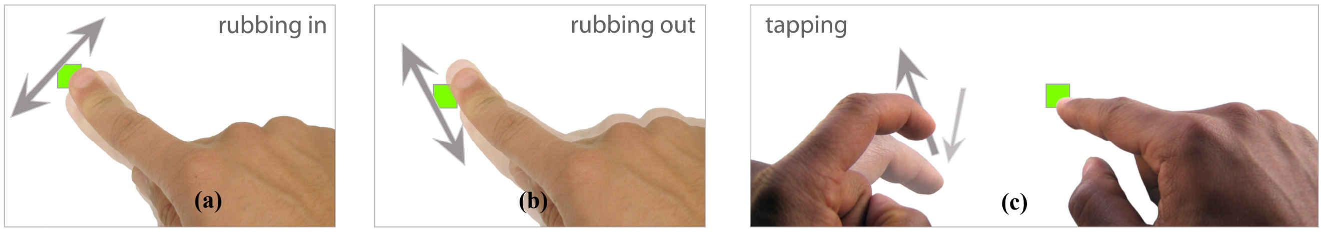

Figure 1. Rubbing and tapping gestures activate operations while the user is touching the display, so that additional parameter control and functionality can be activated during the fluid interaction. (a) Rubbing in and (b) rubbing out support two operations. (c) Bimanual interaction on single-touch displays is simulated with a set of "tapping" techniques, where operations are executed by tapping with a secondary finger (left), while the primary finger (right) is touching the display.

ABSTRACT
We introduce two families of techniques, rubbing and tapping, that use zooming to make precise interaction on passive touch screens possible. Rub-Pointing uses a diagonal rubbing gesture to integrate pointing and zooming in a single-handed technique. In contrast, Zoom-Tapping is a two-handed technique in which the dominant hand points, while the non-dominant hand taps to zoom, simulating multi-touch functionality on a single-touch display. Rub-Tapping is a hybrid technique that integrates rubbing with the dominant hand to point and zoom, and tapping with the non-dominant hand to confirm selection. We describe the results of a formal user study comparing these techniques with each other and with the well-known Take-Off and Zoom-Pointing selection techniques. Rub-Pointing and Zoom-Tapping had significantly fewer errors than Take-Off for small targets, and were significantly faster than Take-Off and Zoom-Pointing. We show how the techniques can be used for fluid interaction in an image viewer and in existing applications, such as Google Maps.

Author Keywords
Pointing, interaction techniques, touch screens, rubbing, tapping, Rub-Pointing, Zoom-Tapping, Rub-Tapping.

ACM Classification Keywords
H.5.2 (User Interfaces): Graphical user interfaces, Input devices and strategies, Interaction styles.

INTRODUCTION
Passive touch screens and finger-pointing interaction techniques are well established in public installations, such as information kiosks and automated teller machines. Additionally, they are gaining popularity in consumer devices, such as cell phones and PDAs. While passive touch screens are intuitive and easy to learn, there are severe limitations in the precision with which a user can interact with them. While it is easy to select large objects by finger pointing, it can be difficult to select very small objects or specify pixel-accurate locations. This type of interaction can be critical in effective selection of small targets on maps or small GUI elements in an operating system [1, 4, 18]. Touch-screen interaction can be complicated by occlusion of the target by the user's hand, imprecision in selection with a finger that is relatively large compared with the target, poor calibration, and parallax errors caused by the offset between the display surface and the overlaid touch surface.

We are especially interested in touch-screen interaction techniques that support fluid interaction and do not require multiple steps [1] or rely on on-screen widgets [1, 20]. We would like our techniques to, as much as possible, behave like the ones from which they are derived, with the possibility of supporting more precise targeting when necessary, without disrupting the overall interaction. In this paper, we explore how this goal can be achieved by directly integrating zooming gestures with the pointing action. We address applications in which zooming the display is not problematic, and may even be desirable. This includes not only full-fledged zoomable user interfaces [3], but also systems for browsing maps (e.g., [8]), exploring images or navigating other types of information spaces.

We introduce and evaluate two families of interaction techniques, rubbing and tapping, comparing them with two existing baseline techniques. Rubbing and tapping both use zooming to make possible precise interaction on commodity passive touch-screen devices. Tapping takes advantage of unique features of passive touch-screens, whereas rubbing can also be used on devices with active digitizers, such as Tablet PCs.

The rubbing techniques (Figure 1a–b) augment single-handed pointing with integrated gestural interaction to trigger actions. One example is Rub-Pointing, in which rubbing the screen zooms in about the targeted location.

The tapping techniques (Figure 1c) enable two-handed interaction on single-touch displays, allowing the user's non-dominant hand to perform an action in parallel with pointing by the dominant hand. One example is Zoom-Tapping, in which the dominant hand points, while the non-dominant hand zooms by tapping the screen. This is accomplished by taking advantage of the ability of many passive touch-screen technologies to average the location of multiple simultaneous points of contact, such that a second touch can be inferred. (While multi-touch devices and technologies are starting to appear [2, 9, 10, 14, 6], we note that single-touch passive touch-screens are still the most common technology on devices such as public kiosks, point-of-sale displays, personal computers and handhelds.)

We also combine rubbing and tapping to create Rub-Tapping, a hybrid two-handed technique in which rubbing controls zoom, while tapping confirms the selection.

To demonstrate our touch screen techniques in practical applications, we developed TouchView—a zoomable image viewer that avoids the use of on-screen widgets altogether. To enable the use of our techniques in third-party programs, we implemented a daemon for Windows XP that intercepts touch screen data in the background, listening for gestures. When rubbing or tapping is detected, the daemon can send various events to the active window. We demonstrate how we use this to, for example, control zooming and panning in Google Maps in a web browser.

RELATED WORK

Potter and colleagues [18] introduced the Take-Off interaction technique for high-precision touch-screen pointing, in which the user controls a cursor that is located slightly above the finger to ensure its visibility, and the selection is made upon releasing the finger from the surface. Albinsson and Zhai [1] compared Take-Off with traditional Zoom-Pointing and two new touch-screen interaction techniques that do not use zooming. In Zoom-Pointing, a user first enters zoom mode to specify a bounding box that is scaled up to fill the workspace, and then points and selects within the scaled space. Albinsson and Zhai showed that, averaged over all target widths, Zoom-Pointing was significantly faster than the other techniques they tested, with the same error rate.

As Buxton points out [5], a touch screen and stylus with a button or tip-switch offers a three-state input model, supporting an out-of-range state (offering passive tracking, since the finger/stylus is visible relative to the screen), a tracking state (button/tip-switch not depressed), and a dragging state (button/tip-switch depressed). In contrast, many finger-operated touch screens provide only two intrinsic states: out-of-range (offering passive tracking) and tracking (often used for selection/dragging by having the touch invoke a mouse button press). Like other work, rubbing and tapping make possible through software the recognition of additional states beyond the two supported directly in the hardware.

To avoid overloading tracking with selection on intrinsically two-state devices, MacKenzie and Oniszczak [12] describe the use of a lift-and-tap gesture to perform selection on a passive touchpad. Benko and colleagues [4] instead use a rocking and pressing gesture of the tracked finger to trigger selection on a vision-tracked tabletop, since they are able to detect the full contact area in the camera image. Their computer vision system also provides support for true multi-touch interaction, enabling sophisticated techniques for controlling precision using multiple hands, which are unfortunately not applicable to common single-touch displays. The techniques we present here use rubbing and tapping on a single-touch passive touch screen to trigger zooming and selection, and are applicable to traditional single-touch displays, as well as to multi-touch displays.

Vogel and Baudisch present Shift [20], a technique for single-touch displays that addresses the problems of the Take-Off technique not by offsetting the cursor, but by instead introducing a small offset callout that displays a copy of the area under the finger with its cursor. The callout is presented automatically when the finger is determined to obscure a sufficiently small potential target, and, in some variants, the small portion of the display in the callout is zoomed for easier selection. In contrast, rubbing and tapping do not rely on the properties of target objects, but can operate instead on the current position alone (e.g., to zoom the entire scene). Rubbing and tapping also provide explicit control over zoom, which is desirable in many browsing applications. Rubbing could even be combined with Shift to support one-handed user-controlled zooming in the callout.

RUBBING AND TAPPING

Rubbing: Expanding pointing with simple gestures

Rubbing provides additional parameter control through a simple rubbing gesture while interacting with a touch screen. Our approach takes into account the orientation of the gesture, to allow for direct access to two operations by "rubbing in" or "rubbing out." After touching the display, rubbing is performed through a small repetitive diagonal motion of the finger (Figure 1a–b), described in detail below. A right-handed user "rubs in" by rubbing back and forth along the lower-left-to-upper-right diagonal, and "rubs out" by rubbing along the lower-right-to-upper-left diagonal (with the motions switched for left-handed users). We found that rubbing out is slightly less convenient to perform, which gives rise to an effective distinction between the two in kinesthetic memory.

In the implementation used in our study, we mapped "rubbing in" and "rubbing out" to zooming in incrementally in discrete steps and resetting zoom, respectively. Other prototypes, which we describe later, replace the action of resetting zoom with zooming out incrementally. Alternatively, different actions could be invoked, such as increasing or decreasing the level of detail displayed, or cycling through different visualization approaches. (For example, an early implementation of rubbing was used to control the distortion in a fisheye lens [17].)

We developed the rubbing and tapping gestures through an iterative experimental design process. Results from a ten-user pilot study allowed us to reject several complicated designs in favor of diagonal rubbing, which was the most comfortable, easiest to learn and practical. A second five-user pilot study allowed us to fine-tune zoom factors, thresholds, behavior, and timeouts (all default numeric values are, however, run-time modifiable and configurable).

Rubbing is detected as a series of discrete, roughly diagonal, strokes. A short quick stroke, followed by another short quick stroke in the roughly *diagonally opposite* direction identifies rubbing. We recognize strokes and their directions through simple tests. Consider the three successive cursor position samples: ($x0$, $y0$), ($x1$, $y1$), and ($x2$, $y2$). If sign ($x1$–$x0$) ≠ sign ($x2$–$x1$) AND sign ($y1$–$y0$) ≠ sign ($y2$–$y1$), then ($x1$, $y1$) ends the previous stroke and begins the next. The slope of the line between the first and last points of a stroke must be finite and positive for rubbing in, finite and negative for rubbing out.

A stroke must also meet length and time constraints to be considered part of a rubbing action. A rubbing stroke must be longer than three pixels and shorter than fifty pixels and end within 500 ms of the previous stroke. An initial pair of these quick strokes in roughly opposite directions (i.e., strokes whose slopes have the same sign) identifies rubbing and triggers initial zooming. The deliberate nature of the gesture minimizes the risk of the rubbing strokes being accidentally confused with other finger movement on the surface (e.g., for dragging and panning).

Rubbing involves directional gestures, which suggests an interesting comparison to marking and marking menus [11]. However, unlike the marks of marking menus, the tiny strokes of rubbing are scale-dependent (strokes that are too large are ignored), time-dependent (strokes must be completed sufficiently quickly to be counted) and always compound (a pair of strokes is required to initiate the action).

We also investigated the use of a clockwise circular motion to rub in, and a counterclockwise circular motion to rub out, inspired by a long history of earlier work on rotational gestures (e.g., [16, 7, 15, 19]). Rotational gestures take longer to detect, however, requiring the user to perform a relatively precise movement in the form of a full circle to activate the desired action. We also find it disadvantageous that the circular motion seemed to be most naturally performed around the target, in contrast to diagonal rubbing, in which the user points directly to the target and, if desired, rubs through it.

It is worth noting that rubbing actions become increasingly harder to use when targets are very close to the edges of the touch-sensitive surface. This could be addressed in software by taking into account the starting position of the rubbing gesture and adjusting its behavior near edges, much like the edge-awareness of Shift [20].

Tapping: Bimanual input on passive touch screens

To enable support for limited multi-touch interaction, we make use of a common passive touch-screen characteristic, similar to the approach used by Matsushita and colleagues [13]. Many types of passive touch screens, when touched at two locations, will report a cursor position somewhere along the vector between the first and second point. The reported cursor position will typically be closer to the point where most pressure is applied.

When a location is touched, followed by a sufficiently large cursor movement while maintaining the touch, we can suspect a second touch. We can conclude that we are not dealing with dragging if the change in reported position exceeds a fixed velocity threshold—it would most likely be caused by a second touch. A large jump back to the original location then indicates the release of the second touch.

We use the term *tap* to refer to a quick touch-and-release action with a *secondary* finger. Tapping requires that the first touch is not released during tapping with the secondary finger. The primary finger must remain stationary during the touch-and-release of a tap, but can be repositioned between taps.

Tapping can be used to trigger a range of functionality. In its simplest form, tapping detected at any location could trigger an action. For example, this allows touching to be treated only as tracking on a passive touch screen, with tapping mapped to a click. Furthermore, multiple, spatially distinct *tapping zones* can be defined, where each triggers different functionality, such as left and right mouse clicks.

It is important that the tap be performed sufficiently far from the first touch, so that it is not confused with a dragging motion; otherwise, it will not be detected as a second touch. However, the distance at which the second touch is registered depends not only on the distance between the two contact points, but also on the relative pressure applied to each point. We use a threshold distance of 50 pixels (a run-time modifiable parameter determined during our pilot studies) to differentiate a tap from a dragging motion, based on the case of two equally firm touches, with an added safety margin. It is important to note, however, that a failed tap will not have critical consequences: It will merely cause the cursor to move back and forth between the locations. The user can immediately notice that the desired action was not performed and repeat the tap in a less ambiguous fashion.

Reducing the error rate with a click

During a pilot study of our techniques, it became clear that participants sometimes unintentionally triggered a selection by accidentally releasing contact. This can occur quite easily on touch-surface technologies that require that firm pressure be maintained, such as the common resistive touch surface we used. To address this issue, we have explored the option of confirming selection with a separate gesture.

We use the term *click* to refer to a quick touch-and-release action with the *primary* finger following an initial release, the same gesture that MacKenzie and Oniszczak [12] referred to as "lift and tap." To trigger selection with a click, the user must quickly touch and release the finger. If the user touches the surface for longer than a preset timeout, then no select action will be triggered on release. The default timeout was chosen to be 250 ms. The user can then retry by quickly clicking with the finger again. The advantage of this approach is that users are less likely to accidentally trigger a select behavior while the finger is being moved on the surface, as in repositioning or rubbing. However, an accidental touch and release that happens before the preset timeout will be registered as a click and result in an error. We also expected the click to introduce a slight performance penalty, since selection will now require an additional confirming click.

NEW INTERACTION TECHNIQUES

We implemented two rubbing techniques (Rub-Pointing and Rub-Pointing.Click), two tapping techniques (Zoom-Tapping and Zoom-Tapping.Click), and a hybrid technique (Rub-Tapping) within a test application, described below.

Rub-Pointing "Rub to Zoom, Release to Select"

Rub-Pointing (Figure 2) allows the user to touch the display and release the finger to select, similar to Take-Off [18]. However, while touching the display, the user can also execute rubbing gestures: the first pair of rub-in strokes zooms by a factor of two, as does each successive single rub-in stroke, and the first pair of rub-out strokes resets the zoom level. As in Take-Off, the user can also adjust their finger position, before selecting the object by lifting the finger off the surface. However, it is important that the user maintain contact with the surface while rubbing to avoid accidental selection. Rub-Pointing makes it possible to zoom in a fluid direct-manipulation action, and supports repeated zoom-in and zoom-reset actions.

Rub-Pointing.Click "Rub to Zoom, Click to Select"

Rub-Pointing.Click (Figure 3) is a version of Rub-Pointing that requires a final confirming click for selection to reduce errors.

Zoom-Tapping "Tap to Zoom, Release to Select"

Zoom-Tapping (Figure 4) allows the user to touch the display with a primary finger, and, if desired, tap with a secondary finger to zoom in. A selection is performed when the primary finger is released.

Each tap zooms in further around the location specified by the primary finger, with a default magnification factor of four for each tap. The user can thus quickly gain additional precision with a few taps and select the target of interest by lifting the primary finger. Our experiments and pilot studies indicated that a larger zoom factor was suitable for tapping, than for rubbing, given its better control over simultaneous pointing (with the stationary primary finger) and zooming (through tapping with the secondary finger).

The advantage of Zoom-Tapping is that it is simple to learn and provides a quick means for magnification. In the interest of keeping the interaction simple for study participants, our implementation maps only one function (zoom in) to tapping, thus making it impossible to reset the zoom level. A straightforward extension would employ the previously mentioned tapping zones (e.g., zooming in with a tap in the top half of the screen, and resetting zoom with a tap in the bottom half).

Zoom-Tapping.Click "Tap to Zoom, Click to Select"

Zoom-Tapping.Click (Figure 5) is a version of Zoom-Tapping that requires a final confirming click for selection to reduce errors.

Rub-Tapping "Rub to Zoom, Tap to Select"

Rub-Tapping (Figure 6) combines rubbing and tapping to create another way to avoid accidental selection caused by an unintended release. Rubbing is used to zoom in or reset zoom (as in Rub-Pointing), but a tap with the secondary finger is required to confirm the selection while the primary finger remains on the screen. In contrast to the techniques that confirm a selection with a click, Rub-Tapping requires that two fingers simultaneously touch the screen to complete the selection (which should not occur otherwise).

EVALUATION

To evaluate our techniques, we used our test application to conduct a formal user study of Rub-Pointing, Rub-Pointing.Click, Zoom-Tapping, Zoom-Tapping.Click, and Rub-Tapping, comparing them against two baseline techniques: Take-Off and Zoom-Pointing.

Apparatus

The experiment used a dual 3.0 GHz Pentium Xeon PC running Windows XP with a 15" resistive touch screen display (MultiQ MQ 158 POS). The display's 1024×768 native XGA resolution results in a pixel triad width of ap-

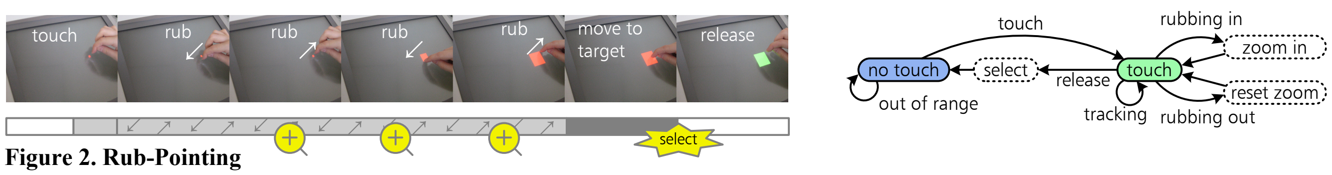

Figure 2. Rub-Pointing

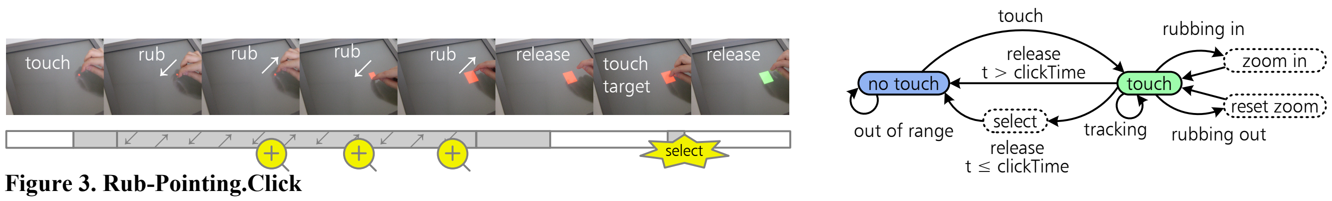

Figure 3. Rub-Pointing.Click

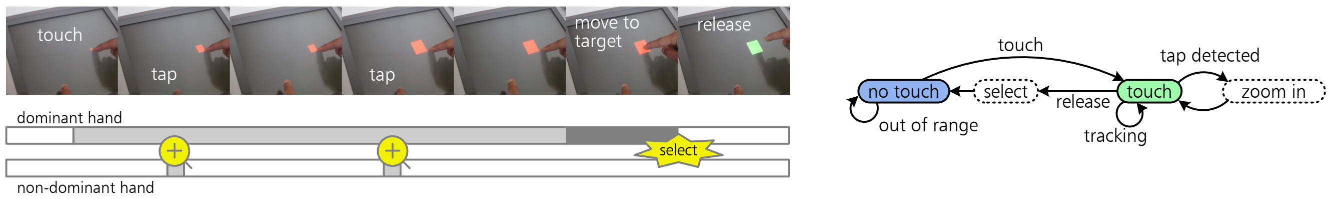

Figure 4. Zoom-Tapping

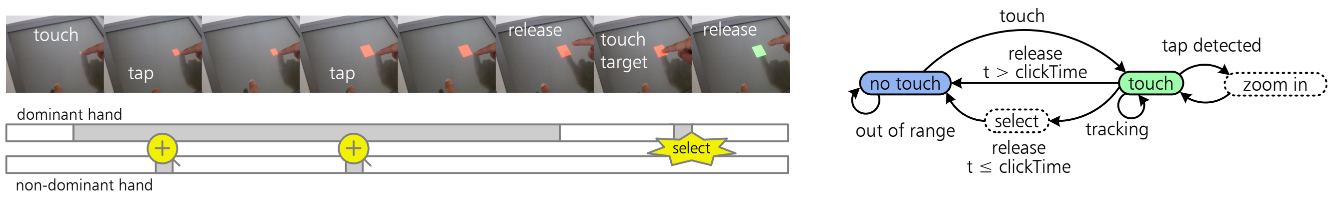

Figure 5. Zoom-Tapping.Click

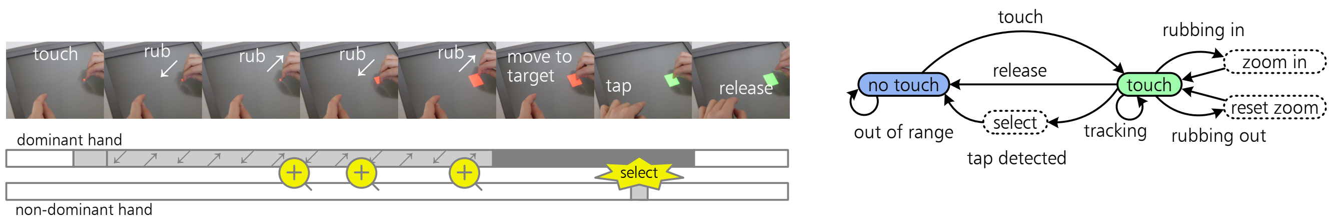

Figure 6. Rub-Tapping

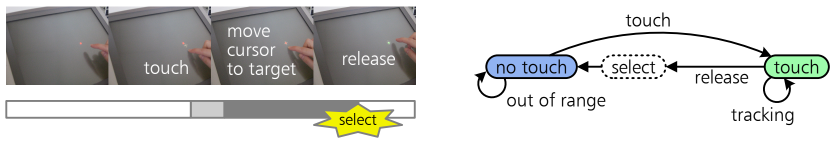

Figure 7. Take-Off (baseline technique)

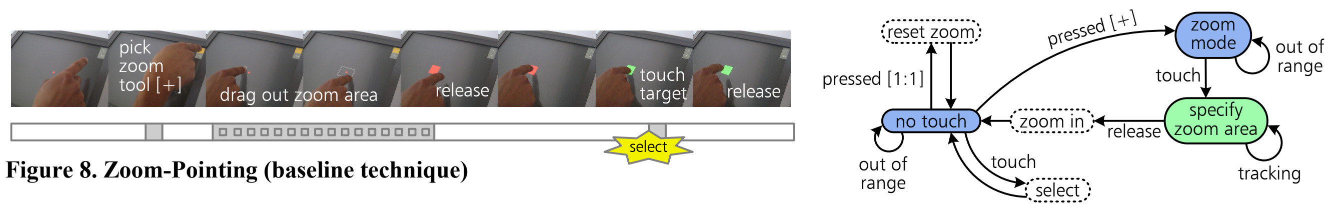

Figure 8. Zoom-Pointing (baseline technique)

For clarity, targets are colored red before selection and green after selection in Figures 2–8.

Legend

☐	not touching display	■	moving finger/cursor to target	⊕	zoom in!
▨	touching display	▨	rubbing	✸	select target!
		▫▫▫	dragging out zoom rectangle		

proximately 0.3 mm and the display was tilted approximately 15° backwards for user comfort. The experimental software was implemented with OpenGL and C++.

Participants

Twenty right-handed volunteers participated in the study and each received two cinema tickets as compensation. The 8 female and 12 male participants were between 19 and 34 years old (average 23.9, standard deviation 3.89). They were, or had previously been, university students. The majority of the participants were students in Media Technology at the Royal Institute of Technology. All participants had used touch screens in public information kiosks, such as ticketing machines for public transportation. None had previously seen or been exposed to our rubbing and tapping techniques. Seven used touch screens often, whereas thirteen had only used them a few times. Prior to performing the study, most participants were positive about using touch screens of the size used in our study, with a few neutral participants and one negative participant. Four participants mentioned that, in their experience, such touch screens often are not sufficiently sensitive and require hard presses.

Baseline 1: Take-Off "Release to Select with Offset Cursor"

As a first baseline for comparison, we implemented a version of the well-known Take-Off technique (Figure 7), which has also been used as a baseline in previous studies [18, 1, 4, 20]. Take-Off is the only technique that we tested that does not provide zooming capabilities, which makes it harder to select small targets. Due to the large size of the finger and the risk of occluding the cursor, Take-Off places the cursor at a fixed offset above the finger. After touching the display, the user can adjust the cursor position until it is over the target, and then select by releasing the finger (as shown in Figure 7). Additional drawbacks of this approach are the difficulty of knowing exactly where the cursor will appear upon contact and the inability to select objects at the bottom of the screen in an unmodified implementation.

Baseline 2: Zoom-Pointing "Use Tool to Zoom, Click to Select"

Our second baseline was Zoom-Pointing (Figure 8), implemented to be as close as possible to the version described by Albinsson and Zhai [1], in which it consists of a button that activates the zoom mode, and a button that resets the zoom level. Zoom-pointing is a tool-based technique that is common in many modern graphical applications. If the target is sufficiently large, the user can touch it right away. Alternatively, the user can enter a zoom mode by touching the zoom button with the finger. After the finger has been released, the user can specify a zoom area by drawing out a rectangle with the finger. The application zooms up the screen to this rectangle after the finger has been released. The user can then select the target by touching it, touch the zoom button to enter zoom mode again (if the target is not sufficiently large), or reset the zoom level by touching the reset button (e.g., if the desired target zoomed off the screen). In contrast to the other techniques, Zoom-Pointing selects on touch, not on release, making it more error prone: the user cannot adjust the position or invoke zoom after touching. It also uses a different interaction style, involving multiple target acquisition steps (zoom button, rectangle, target), that breaks up the desired fluid interaction, as illustrated by the state chart in Figure 8.

Procedure

Each participant was asked to alternately select two targets placed 250 pixels apart, well away from the edges of the screen, in a reciprocal 1D pointing task, where zoom level was reset after each target selection. To maximize contrast, targets were green squares of varying size on a black background. A large grey offset rectangular outline helped the participant identify the position of the target at the beginning of each trial. The rectangular outline was hidden upon touch. Auditory feedback was provided with a low-frequency beep for errors and a high-frequency beep when the participant successfully selected a target. The software logged times and hit positions, such that completion times and error rates could be derived.

Design

A repeated-measures, within-subjects study was performed. There were five target widths (1, 2, 4, 8, and 16 pixels = 0.3, 0.6, 1.2, 2.4 and 4.8 mm) and seven techniques (Take-Off, Zoom-Pointing, Rub-Pointing, Rub-Pointing.Click, Zoom-Tapping, Zoom-Tapping.Click, and Rub-Tapping). The order in which the techniques were presented was randomized, and the order in which sizes were presented was randomized for each block of trials. An analysis found no significant effects of order on the results. After seeing a demonstration of a technique, a participant performed an initial block of 10 practice trials (2 trials × 5 widths) with that technique, where each trial needed to result in a successful selection for the program to proceed, to ensure that the participant experienced successful selections for the technique. However, the participant was allowed to ask the experimenter to manually advance the test to the next trial after an unsuccessful attempt for difficult conditions, such as Take-Off with 1-pixel targets. This first block was followed by a block of 15 practice trials (3 trials × 5 widths) for the technique that behaved as in the real test, where a failed attempt would always proceed to the next trial. After this preparation, the participant performed a block of 70 test trials for the technique (14 trials × 5 widths). Consequently, we had:

	1 trials	×	5 widths	=	5 demonstration trials
	2 trials	×	5 widths	=	10 practice trials (must succeed)
+	3 trials	×	5 widths	=	15 practice trials
+	14 trials	×	5 widths	=	70 test trials
					95 trials
				×	7 techniques
					665 selections per participant

Thus, for each of the seven techniques, a participant received a demonstration, followed by two blocks of practice trials, followed by a block of 70 test trials. We found it necessary to group the demonstration, practice trials, test trials and questionnaire together for each technique to avoid confusing the participants, given the large number of techniques tested and the similarities in how they worked.

Hypotheses

Prior to running the experiment, we formulated the following hypotheses:

H_1: For small targets, Take -Off will have higher error rates than all other techniques, due to its lack of support for zooming.

H_2: As target size increases, Take-Off will approach the error rates of the rubbing and tapping techniques.

H_3: As target size increases, Take-Off will approach the completion-time performance of the rubbing and tapping techniques.

H_4: Rub-Pointing.Click and Zoom-Tapping.Click will have fewer errors than Rub-Pointing and Zoom-Tapping, respectively, due to the added click timeout.

H_5: Zoom-Tapping and Zoom-Tapping.Click will be faster than Rub-Pointing and Rub-Pointing.Click, respectively, for small targets, since each rub moves the targeting location and requires reacquisition of the target after zoom.

H_6: Rub-Tapping will have the best error rate because it is the only technique where selection requires a simultaneous tap with the secondary finger for selection (which would be unlikely to occur by accident).

Analysis

To mitigate the common skewing associated with human response times, and remove the influence of potential outliers, we analyzed the median (rather than mean) completion times for each block of 14 repetitions per cell (14 trials × 5 widths × 7 techniques per subject). A within-subjects analysis of variance (ANOVA) was then performed on the median completion times. In addition, a within-subjects ANOVA was performed on the mean error rate for all techniques over all target sizes. We used an α of 0.05 to determine statistical significance.

Error rate

A within-subjects ANOVA of mean errors show that target size ($F_{4, 76} = 34.22$, $p < 0.001$) and technique ($F_{6, 114} = 24.74$, $p < 0.001$) had a significant effect on error rate, with a significant interaction between size and technique ($F_{24, 456} = 24.39$, $p < 0.001$). Paired samples t-tests with a Bonferroni adjustment show that Take-Off had significantly more errors than all other techniques for 1-pixel targets. It was also significantly worse than all but Zoom-Pointing for 2-pixel targets. Finally, for 4-pixel targets, it was significantly worse than Rub-Pointing.Click.

Zoom-Pointing had significantly more errors than all rubbing techniques (Rub-Pointing, Rub-Pointing.Click, and Rub-Tapping) for 1- and 2-pixel targets. It also had significantly more errors than Zoom-Tapping.Click for 2-pixel targets.

No significant difference was found in error rate between the other techniques across the different target sizes and we could thus not verify hypotheses H_4 and H_6.

These results confirm hypotheses H_1 and H_2. Figure 9 clearly illustrates how the error rate for Take-Off decreases with increased target sizes. A mean error rate of 3.5–14% for our techniques is shown, where Rub-Pointing.Click (3.5–5.7%) and Rub-Pointing (6.4–11%) have the best results.

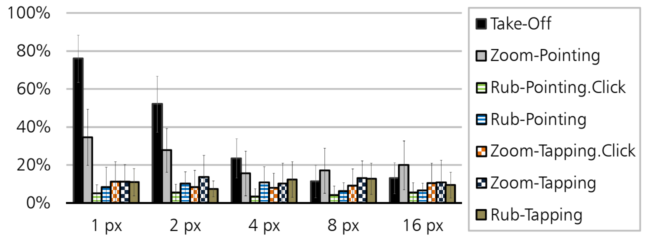

Figure 9. Mean Error Rates and SEM (Standard Error of the Mean) for the seven techniques at different target sizes.

Completion time

Similar to the experience of Benko and colleagues [4], we had two blocks without a single completed trial for the smallest targets. These blocks happened for two of the participants in the difficult condition of Take-Off with 1-pixel targets. We therefore chose to divide our analysis of completion times into two parts.

First, we conducted an ANOVA on target sizes 2–16 pixels over all techniques, which showed that technique had a significant effect on completion time ($F_{6, 114} = 84.67$, $p < 0.001$). Paired samples t-tests with a Bonferroni adjustment show that Take-Off was significantly slower than all our rubbing and tapping techniques for 2-, 4- and 8-pixel targets. Rub-Pointing was significantly faster than Take-Off for 16-pixel targets ($t_{19} = 5.96$, $p < 0.001$). This result confirms hypothesis H_3 with the exception of Rub-Pointing being significantly faster than Take-Off for all target sizes.

Zoom-Pointing proved to be significantly slower than all rubbing and tapping techniques over all sizes, and significantly slower than Take-Off for the 16-pixel targets ($t_{19} = 5.21$, $p < 0.001$), as shown in Figure 10.

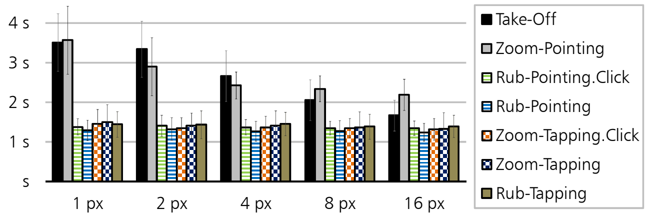

Figure 10. Mean Median Completion times and SEM (Standard Error of the Mean) for the seven techniques at different target sizes.

Second, we ran an ANOVA on 1-pixel targets over all techniques except Take-Off, and found a significant effect of technique on completion time ($F_{5, 95} = 103.33$, $p < 0.001$). Paired samples t-tests with Bonferroni adjustment showed that Zoom-Pointing was also significantly slower than the other techniques for 1-pixel targets.

No significant differences were found between the rubbing and tapping-techniques at any target size, and hypothesis H_5 could thus not be confirmed.

Qualitative Feedback

After finishing the trials for a technique, the participants filled in a questionnaire in which they ranked the experienced technique on a seven-point Likert scale (–3 to 3) with regard to learnability, ease of use, comfort, user experience, and perceived speed. They were also encouraged to provide written and verbal comments. At the end of the study, they commented on the techniques they liked the most and the least. A summary of the results is shown in Figure 11. The experimenter also took notes, recording events of interest that occurred during the test.

Learnability

Several participants commented that all the techniques were easy to understand after instruction.

Ease of use

Especially for small targets, there were frequent comments that Take-Off was "impossible to use", while some participants commented that it was easy to use for large objects. While Rub-Pointing.Click seemed easier to use than Rub-Pointing, this was not the case for Zoom-Tapping.Click and Zoom-Tapping. The experimenter noticed several cases where the participant would forget to assure that the position of the finger was over the target for the non-click versions, rendering Rub-Pointing.Click more robust than Rub-Pointing. For Zoom-Tapping.Click, however, the confirming click could result in confusion, since the actions for the dominant and non-dominant hands became similar. Three participants commented that it was not intuitive to activate selection by tapping *outside* the target for Rub-Tapping.

Comfort

Many participants complained that Take-Off demanded excessive visual concentration, which led to eye fatigue. Visual concentration alone was often not sufficient, and the participant would lean over the display, resulting in shoulder and back strain.

The major disadvantage of the tapping techniques was the requirement of two hands, which participants felt was more complicated and less comfortable. The use of two hands also appeared to require higher cognitive load to coordinate them, which seemed to be especially evident in Zoom-Tapping.Click, where the actions of the dominant and non-dominant hands were sometimes confused.

The combination of rubbing techniques and the display surface resulted in friction, which was uncomfortable for some participants. Several commented that a display with less friction would work better. Surface acoustic wave touch screens (e.g., as used by Albinsson and Zhai [1]) are typically made of glass and have a more sensitive touch response, reporting both location and a measure of applied pressure. They are, however, less common in consumer devices than resistive touch screens, such as the one we used. Rub-Tapping generated friction from rubbing *and* required two hands, and was disliked by most participants.

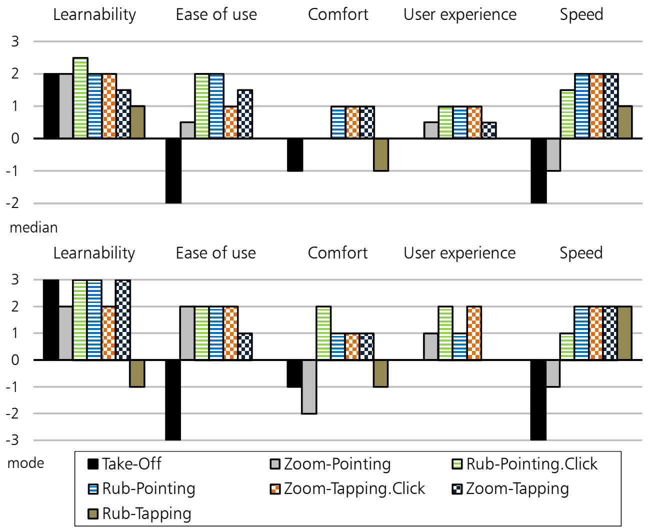

Figure 11. The seven techniques, as rated by the participants. *Top:* Median plot. *Bottom:* Mode plot.

User experience

Take-Off was frustrating due to the targeting difficulties for small objects, which annoyed several participants. Participants found Zoom-Pointing cumbersome, due to the many steps involved in the interaction.

Speed

Take-Off, which was perceived to be the slowest, demanded a great deal of concentration for smaller objects, whereas Zoom-Pointing, perceived to be the second slowest, required multiple steps.

Most preferred techniques

Participants were asked to specify and comment on the techniques they liked the most. Zoom-Pointing ("natural", "intuitive", "no need to rub"), Rub-Pointing ("fast"), Rub-Pointing.Click ("precise"), and Zoom-Tapping.Click ("precise") were each selected by four participants. Zoom-Tapping ("fast") was selected by five participants. (One participant specified both versions of Zoom-Tapping as the most preferred and one participant specified both Rub-Tapping and Rub-Pointing.) No one selected Take-Off. The most popular properties cited in the comments (by eight participants each) were intuitiveness and speed. None of the participants that preferred Zoom-Pointing mentioned speed, however.

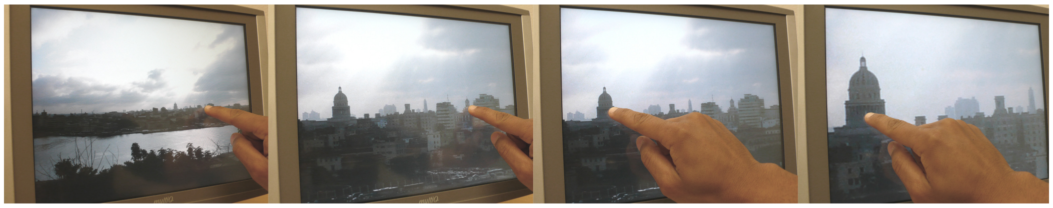
Figure 12. Rubbing in the TouchView image viewer allows fluid zooming, integrated with pointing and panning, while avoiding the need for on-screen widgets.

Least preferred techniques

Participants were also asked to specify and comment on the techniques they liked the least. Ten participants chose Take-Off ("difficult to target"), whereas five picked Zoom-Pointing ("too many steps", "tedious to select the zoom tool"). Two participants selected Rub-Tapping ("slow", "not logical"). Two participants disliked all the rubbing techniques ("uncomfortable"). The two participants that disliked Zoom-Tapping.Click ("confusing") had Zoom-Pointing as their favorite ("easy").

Discussion

As in previous studies [1, 4], Take-Off had a high error rate for small targets and was increasingly better for larger targets, approaching the performance of our rubbing and tapping techniques. However, given Take-Off's poor performance for small objects, it was listed as the least preferred technique by half of the participants.

Zoom-Pointing, which was significantly faster than Albinsson and Zhai's techniques [1] was significantly slower than all our techniques. The numerous sequential steps were not only perceived as cumbersome and slow, but also require visual attention and provide multiple opportunities for mistakes, such as not pressing sufficiently hard when dragging out the zoom area or forgetting to activate the zoom button in repeated zooming.

The study shows that all rubbing and tapping techniques were significantly faster than Take-Off for up to 8-pixel targets (2.4 mm) and faster than Zoom-Pointing for all sizes. They also had fewer errors than Take-Off and Zoom-Pointing for small targets. There were no significant differences in speed between the rubbing and tapping techniques. The variations of Rub-Pointing and Zoom-Tapping that used an additional click were not significantly slower, which is encouraging.

The major disadvantage of the rubbing techniques was the fatigue incurred by the friction between the finger and the display. However, they had a high perceived speed and ease of use. It is interesting to note that some users did not like the tapping techniques because they required two hands and were more attention-demanding than rubbing, even in the case of the single tap for Zoom-Tapping.

Our results also supported the importance of having a distinct separation of operations, whether in a single-handed gesture (e.g., rubbing in and rubbing out) or a bimanual interaction (e.g., Zoom-Tapping). Similarity between the dominant hand click and the non-dominant hand tap for Zoom-Tapping.Click, for example, confused several users, and led to a higher error rate

Finally, Rub-Tapping provided no advantage, as there was no significant difference in error rate—it merely combined the disadvantages of the rubbing and tapping techniques.

TOUCHVIEW: A ZOOMABLE IMAGE VIEWER

As an example of a practical application, we developed TouchView, a touch-screen display photo viewer that uses rubbing and tapping (Figures 12–13). TouchView was implemented in C++ and OpenGL. The user can pan the image with a finger, using familiar dragging motions, and control the zoom level, using rubbing or tapping. In TouchView, we use two tapping zones, where tapping in the upper half of the display zooms in, and tapping in the bottom part zooms out. TouchView demonstrates how our techniques allow a clean and simple passive touch-screen interface that supports panning and zooming, but avoids the use of on-screen widgets that might otherwise occlude content of interest.

TOUCHDAEMON: A PLUGIN TO ENABLE RUBBING AND TAPPING IN EXISTING APPLICATIONS

There are many applications for which we do not have control over their source code or cannot otherwise modify their behavior. To address these, we developed TouchDaemon, a program that can be run in the background under Windows XP to detect rubbing and tapping gestures being performed on a touch-screen display. TouchDaemon uses Windows hooks along with the Raw Input Model to detect and inter-

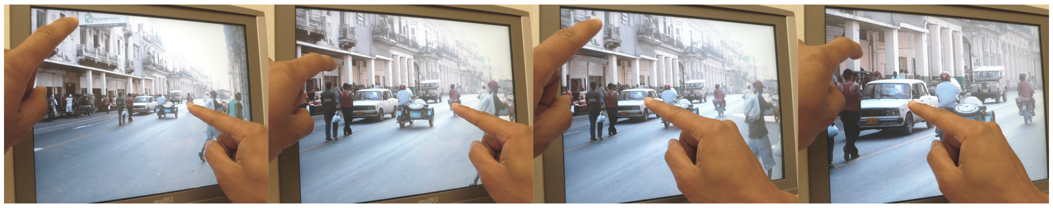
Figure 13. TouchView allows the user to incrementally zoom in and out using two tapping zones. Here, the user taps in the upper part of the screen to zoom in on the location indicated by the right hand finger.

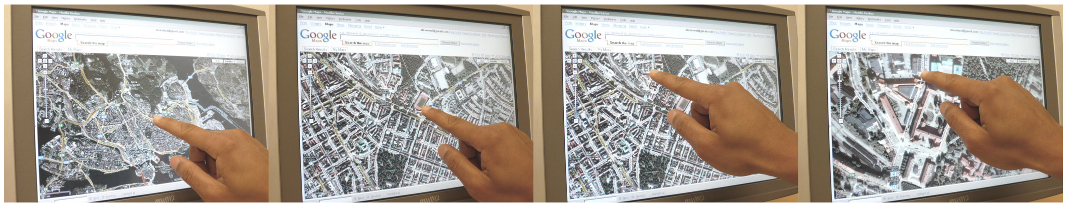

Figure 14. TouchDaemon allows rubbing and tapping to control unmodified Windows applications, such as Google Maps running inside a web browser.

cept touch-screen events. Upon recognizing rubbing and tapping gestures, it can replace them with appropriately mapped actions.

As shown in Figure 14, we use rubbing to control Google Maps [8], running in a web browser. We accomplish this by mapping rubbing gestures to scroll events in TouchDaemon. When the unmodified JavaScript on the web page receives the synthesized scroll-up or scroll-down event, it initiates a zoom-in or zoom-out action.

CONCLUSIONS AND FUTURE WORK

We have introduced and evaluated two families of single-handed and bimanual interaction techniques for interaction on single-touch displays. Given potential parallax issues and occlusion by the user's hand, these techniques avoid relying on widgets and other on-screen visual artifacts, and instead provide precision through simple gestures. We believe this is one of the reasons why our techniques have proven to have consistent performance and high selection speeds (on average < 2 s) for all target sizes.

Our study shows that our techniques perform significantly better than the well-known Take-Off and Zoom-Pointing techniques. The rubbing and tapping techniques combine pointing and zooming into a single direct-manipulation action (in the spirit of combining marking and direct manipulation [11]), and maintain the possibility of direct and simple touch-screen pointing for large objects, while affording the ability to zoom if additional precision is needed.

Based on the results of our studies, and our own experience, we can make a number of recommendations. We suggest using tapping when more control is desired, bimanual interaction is acceptable, and the screen is sufficiently large (due to the distance needed between the two touches for disambiguation). We suggest using rubbing when friction isn't a problem (e.g., a smooth glass screen, as on the iPhone), a single finger is preferred, and the screen is either small (e.g., a cell phone) or large. For increased robustness, a confirming click can be added; our study shows no significant speed penalty despite the additional step.

We intend to evaluate our techniques on smaller displays where precise pointing is even harder. Experiments indicate that our rubbing techniques work well with pen interaction (e.g., on Tablet PCs and passive PDA screens), and we expect that the relative lack of friction in comparison with finger rubbing will eliminate concerns about fatigue.

ACKNOWLEDGMENTS

We thank Johan Wirmark for running an earlier pilot user study, and Ed Cutrell, Hrvoje Benko, and Patrick Baudisch for providing useful feedback and comments. This work was funded in part by NSF Grant IIS-01-21239.

REFERENCES

1. Albinsson, P-A. and Zhai, S. High precision touch screen interaction. Proc. CHI '03, ACM Press (2003), 105–112.
2. Apple iPhone. http://www.apple.com/iphone/. January 2008.
3. Bederson, B., Hollan, J., Perlin, K., Meyer, J., Bacon, D., and Furnas, G. Pad++: A zoomable graphical sketchpad for exploring alternate interface physics. Journal of Visual Languages and Computing, 7(1), March, 1996, 3–32.
4. Benko, H., Wilson, A., and Baudisch, P. Precise selection techniques for multi-touch screens. Proc. CHI 2006, ACM Press (2006), 1263–1272.
5. Buxton, W. A three-state model of graphical input. In D. Diaper et al. (eds.), Human-Computer Interaction— INTERACT '90. Amsterdam: Elsevier Science Publishers B.V. (North-Holland), 449–456.
6. Dietz, P.H., Leigh, D.L. DiamondTouch: A multi-user touch technology. Proc. UIST '01, ACM Press (2001), 219–226.
7. Evans, K. B., Tanner, P. P., and Wein, M. Tablet-based valuators that provide one, two, or three degrees of freedom. Proc. SIGGRAPH '81. ACM Press (1981), 91–97.
8. Google Maps. http://maps.google.com/. January 2008.
9. Han, J. Y. 2005. Low-cost multi-touch sensing through frustrated total internal reflection. Proc. UIST '05. ACM Press (2005), 115–118.
10. Jazzmutant Lemur. http://www.jazzmutant.com/lemur_overview.php. January 2008.
11. Kurtenbach, G. and Buxton. W. Issues in combining marking and direct manipulation techniques. Proc UIST 91, ACM Press (1991), 137–144.
12. MacKenzie, I.S. and Oniszczak, A. A comparison of three selection techniques for touchpads. Proc. CHI '98, ACM Press (1998), 336–343.
13. Matsushita, N., Ayatsuka, Y., and Rekimoto, J. Dual touch: a two-handed interface for pen-based PDAs. Proc. UIST '00. ACM Press (2000), 211–212.
14. Microsoft Surface. http://www.microsoft.com/surface/. January 2008.
15. Moscovich, T. and Hughes, J. F. Navigating documents with the virtual scroll ring. Proc. UIST '04. ACM Press (2004), 57–60.
16. Newman, W. A graphical technique for numerical input. The Computer Journal, 11(1), 1968, 63–64.
17. Olwal, A. and Feiner S. Rubbing the fisheye: Precise touch-screen interaction with gestures and fisheye views. Conf. Supplement of UIST '03 (2003), 83–84.
18. Potter, R.L., Weldon, L.J., and Shneiderman, B. Improving the accuracy of touch screens: An experimental evaluation of three strategies. Proc. CHI '88, ACM Press (1988), 27–32.
19. Smith, G. and schraefel, mc. The radial scroll tool: Scrolling support for stylus- or touch-based document navigation. Proc. UIST 2004, ACM Press (2004), 53–56.
20. Vogel, D. and Baudisch, P. Shift: a technique for operating pen-based interfaces using touch. Proc. CHI '07. ACM Press, (2007), 657–666.