Contact Area Interaction with Sliding Widgets

Tomer Moscovich
Microsoft Research-INRIA Joint Centre
Orsay, France
tomer@moscovich.net

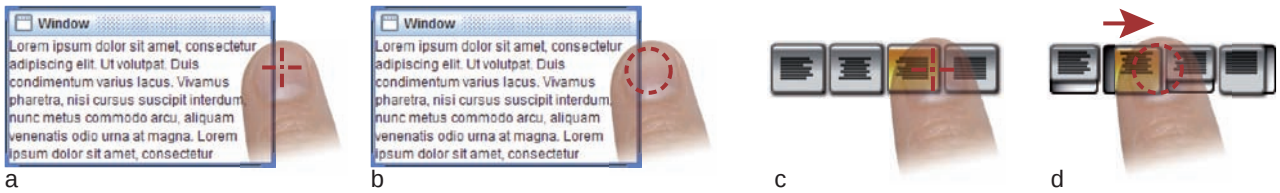

Figure 1: (a) The resize widget is difficult to select since the finger occludes both the target and the selection point. (b) Using a selection area increased the widget's effective width, making it easy to capture. (c) Pushbutton selection is hard to predict when a finger touches more than one button. (d) Sliding Buttons resolve contact ambiguity by varying their sliding direction.

ABSTRACT
We show how to design touchscreen widgets that respond to a finger's contact area. In standard touchscreen systems a finger often appears to touch several screen objects, but the system responds as though only a single pixel is touched. In contact area interaction all objects under the finger respond to the touch. Users activate control widgets by sliding a movable element, as though flipping a switch. These Sliding Widgets resolve selection ambiguity and provide designers with a rich vocabulary of self-disclosing interaction mechanism. We showcase the design of several types of Sliding Widgets, and report study results showing that the simplest of these widgets, the Sliding Button, performs on-par with medium-sized pushbuttons and offers greater accuracy for small-sized buttons.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Input devices and strategies.

General terms: Design, Human Factors.

Keywords: Touch input, touchscreen, interactive surfaces.

INTRODUCTION
The graphical control widgets found on today's touchscreens were developed for mouse- and cursor-based systems with very different input properties than those of touchscreens. Cursor-mediated input offers separate tracking and dragging states, an unobstructed view of the screen, and precise control of a one-pixel selection point. Touchscreens have none of these qualities, making it hard to manipulate standard control widgets. Yet touchscreens possess a number of valuable qualities that are ignored by existing widget designs. Most prior work has attempted to improve touchscreen interaction by making touchscreen input more like mouse input [23, 3, 26, 7]. We propose an alternative: instead of adapting the input to the control widgets, we adapt the control widgets to the input.

We present a new strategy for touchscreen interaction based on area selection and sliding. Our approach approximates a finger's contact area with a small selection region, rather than a single pixel. This representation of touch has two important benefits. First, it resolves the ambiguity regarding which screen object the finger is touching: every object under the fingertip responds to the touch. Second, as with area cursors, contact area selection facilitates small target acquisition by increasing a target's effective width [13, 29].

The need for large, well-spaced controls severely limits the design of touchscreen interfaces. Even well designed interfaces can be frustrating to use when presented on durable public terminals that are prone to parallax errors. The one-pixel selection model is a poor match for users' perception of touch. By making every widget in contact with the fingertip respond to touch, we shift the burden of disambiguating the selection from the user to the interface designer. We propose the use of directional sliders as a means of deciding ambiguous selections. Sliding mechanisms, such as scroll bars and volume faders, are familiar elements of the graphical interface. They use the physical metaphor of a movable handle constrained to a slot to visually indicate how far, and in which direction, the handle can slide. Combining contact area selection with directional sliders is not just a means of eliminating selection ambiguity; it creates a rich new design space of Sliding Widgets that can pack multiple functions into a small space. Furthermore, this framework allows designers to transform symbolic interaction techniques, such as gesture- and mark-based methods, into direct manipulation techniques by providing a physically-inspired visual prompt.

In this paper, we illustrate the design possibilities of Sliding

UIST'09, October 4–7, 2009, Victoria, British Columbia, Canada..

Widgets with several examples. We also report early empirical results which indicate that contact area selection of Sliding Widgets holds promising solutions for touchscreen interface design.

RELATED WORK

Many researchers have recognized the issues associated with touchscreen interaction using point selection. These issues are collectively known as the fat finger problem. They include the ambiguity that arises when a finger touches two or more targets simultaneously, occlusion of both the target and the selection point, and user uncertainty regarding the mapping from the contact area to the selection point. The latter problem is often worsened by parallax issues [17].

One approach to addressing these issues is to resolve the problem of occlusion. The Offset Cursor, proposed by Potter et al. [23] places the selection point slightly above the contact area. A similar offset strategy is used by Albinsson and Zhai's Precision Handle [3], as well as the Windows Vista Touch Pointer [18]. Roudaut et al.'s MagStick ensures the selection point's visibility by mapping it to a position opposite the initial contact [25], while Benko et al. allow users to control cursor offset using an additional finger contact [8]. Selection point offset is not the only means of resolving occlusion-caused ambiguity. Vogel and Baudisch's Shift technique uses a small callout that provides a view of the hidden contact area. Occlusion can be avoided entirely by accepting touch input on the back side of the device [27, 7]. Most of the above techniques use a take-off selection model, which does not support dragging. Contact area-based selection does away with the need for a tracking state for simple selection, and supports dragging as its primary state. While occlusion avoidance techniques play an important role in making legacy applications accessible on touchscreens, they introduce a layer of indirection between the user's touch and screen objects. The finger manipulates a selection widget, or cursor, which in turn controls screen objects. Sliding widgets remove this indirection, allowing users to control objects with direct touch.

An alternative approach to reducing selection point ambiguity is to precede pointing by a zooming step [3, 8, 19]. Users first zoom in on the target using a tool or gesture, and then select it. Some systems zoom in automatically when they detect selection ambiguity [20]. Although these methods effectively reduce occlusion and selection point ambiguity, they make target selection a two step process, an undesirable complication for such a common operation.

While resolving selection point ambiguity is important for supporting legacy systems, Yatani et al. [30] show that it is only one of many methods for resolving *target selection* ambiguity. Their Escape technique uses directional gestures to distinguish one of several nearby targets. This disambiguation method is very similar to that of Sliding Widgets, and is the inspiration for our Control Bead widget. The key difference is that by using the metaphor of real-world manipulation the mechanism for selecting Sliding Widgets becomes self-disclosing. Once the sliding model is understood, it allows novice users to discover software functionality and to decode the operation of novel widgets. In contrast, gesture-based methods must be explained by a manual or a tutorial. While some mark-based techniques such as Marking Menus [16] do have a functionality-disclosing novice mode, the operation of the menu must still be explained to the user. We expect that many mark- and gesture-based techniques will have useful sliding-based analogs, which may be better suited to touch input systems where a command-mode switch is not available.

Using rich shape information for interaction is not a new idea. Krueger's VIDEOPLACE system uses a silhouette of the user's hands to interact with digital objects [15]. Rekimoto describes a technique that uses contact area shape to create a simulated potential field that can be used to manipulate objects [24]. Cao et al. [9] use contact shape, along with surface motion vectors, to manipulate objects in a physically inspired manner, while Wilson et al. [28] use shape and motion input to manipulate objects in a real-time physics simulation. The idea of using contact shape together with a physical manipulation analogy can be seen as a general case of our interaction framework, which only uses a finger contact area approximation along with sliding. However, these contact-shape based techniques are aimed at generic object manipulation, rather than at using simple control widgets, and do not address selection ambiguity. Furthermore, they require much richer input than is available on most touchscreens. We believe that Sliding Widgets will fit well within the context of these general shape-based manipulation frameworks, and will allow them to take advantage of the expressive symbolic language of widget-based interfaces.

Control widgets that contain a sliding element for direct manipulation are common in many interfaces. These are generally used for adjusting continuous parameters, as with sliders and scroll-bars. Sliding has also been used for discrete control, as with the slider-based toggle-switches of Plaisant and Wallace [22] or the unlock slider of the Apple iPhone [5]. The utility of direct manipulation sliding gestures has been recognized as an important element of touch interface design: the interface of Karlson et al.'s LaunchTile [14] is almost entirely sliding-based. Existing Sliding Widgets have proved to be very useful, but we believe their expressiveness and utility for touch interaction can be greatly increased when they are combined with contact area selection. We illustrate this with a number of examples.

CONTACT AREA INTERACTION WITH SLIDING WIDGETS

The width of the human finger is often cited as a cause of difficulties in touchscreen interaction. We show how to design interfaces where the finger's width is an asset rather than a limitation. Figure 1(a,b) shows a simple example. While the window's resize widget is difficult to hit with a one-pixel selection point, using the contact area under the finger makes this task much easier. For area-based selection to behave in a consistent, unambiguous manner, all active objects in the selection area must respond to the touch. Figure 2 illustrates another advantage of this interaction model. The left and right volume sliders can each be adjusted separately, but by placing a finger so it overlaps both slider knobs the user can move them together. Moreover, we can get these advantages with no change to the hardware. We have found that sim-

ply approximating the contact area with a small circle works very well in practice. The usefulness of contact area selection will likely increase with the availability of touchscreens that report multiple contact points or the size and shape of the contact area.

In general, area-based selection has several advantages over point-based selection:

- It allows easy acquisition of small targets;
- removes ambiguity, as every item under the fingertip responds to the touch;
- is resilient to occlusion, since it is easy to predict what occurs in the occluded region;
- is resilient to parallax errors, since a selection area is likely to cover the target even if shifted slightly;
- is compatible with drag-based interaction widgets;
- allows manipulation of multiple controls simultaneously;
- can take advantage of additional contact information such as size and shape.

The disadvantage of area-based selection is that when a finger touches several control widgets, but the user only wants to manipulate one, an additional disambiguation mechanism is needed. This mechanism may add complexity to the interface and may call for the redesign of many common control widgets.

We propose Sliding Widgets as a means of disambiguating area-based selections and for building a variety of useful interaction instruments. Figure 1(d) shows an example of Sliding Buttons. A Sliding Button consists of a small slider thumb mounted in a short track. To activate the button, a user slides her finger so as to cause the thumb to slide to the opposite side of the track. The finger stroke can start outside the button and does not require much accuracy. As long as a small part of the selection area slides across the button in the indicated direction, the slider thumb will catch and move to the on position in analogy to a standard light switch. While it is likely that the contact area will also slide over adjacent Sliding Buttons, their slider thumbs will not move, as their tracks are perpendicular to the stroke direction.

Apart from disambiguating area-based selection, widgets that are activated by sliding have several advantages over those activated by touch or take-off:

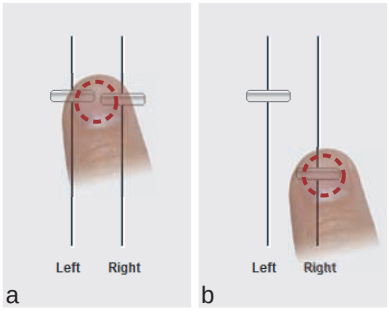

Figure 2: A selection area lets the user control these left and right volume sliders simultaneously (a) or separately (b).

Figure 3: The slider thumb of the OK button can be moved by any touch within the highlighted green area. The Cancel button track is orthogonal to the OK button to avoid accidental activation.

- They pack multiple meanings into a small space by using multiple sliding directions;
- avoid accidental activation by indexical pointing or by brushing against the display;
- provide an additional state analogous to mouse tracking;
- allow mnemonic association of direction with meaning (e.g. back/next buttons);
- support sequential or parallel activation of multiple widgets in a single stroke;
- allow "infinite width" widgets at window edges;
- provide a visible affordance for interpreting novel widgets;
- provide a visual prompt for gesture and crossing methods.

The disadvantage of sliding widgets is that they may require a bit more planning or effort of the user. The user must interpret the sliding directions of the widget and its neighbors to determine the stroke direction, and to avoid activating nearby widgets. We believe that this effort will be small in well designed interfaces, and that the advantages of using contact area selections in combination with Sliding Widgets will outweigh the disadvantages for a significant number of interface designs.

DESIGNING SLIDING WIDGETS

Sliding Buttons

Sliding Widgets are designed for use with contact area selection, so their activation footprint is larger than their visible extent. Figure 3 shows the activation footprint for a simple Sliding Button and a small circular activation area. Any touch within this highlighted area can potentially move the slider thumb. The slider will also move if the selection area slides into this region from above. While this large footprint makes the OK button easy to select, it can also lead to undesirable interactions with nearby widgets. Sliding Buttons with overlapping activation areas must be oriented with either opposing or near-orthogonal sliding directions to avoid ambiguous selections.

Unlike standard touch buttons, where the screen stops the user's finger once a button has been pressed, Sliding Buttons provide no physical feedback when the slider thumb reaches the on position. Thus, it is important to provide alternative forms of feedback. Our Slider Buttons produce a click sound on activation and visually highlight the slider thumb or the button border. To compensate for the fingertip occluding the button, the slider thumb sticks to the on position for a fraction of a second after losing contact with the finger. It then springs back to the start of the track, and the highlighting is disabled.

At each finger motion event, the slider thumb of each Sliding Button under the selection area is advanced or withdrawn by the dot product of the finger's motion vector and the direction vector of the Sliding Button. Thus, if a Sliding Button is brushed by an off-axis stroke aimed at selecting a neighboring widget, the slider thumb will move slightly along the track and spring back when it loses contact. These movements give users feedback as to the degree of error in their strokes, so they may adjust their level of precision. The slider track is kept short relative to the width of the thumb, allowing activation with a short flick gesture, rather than a long stroke. This short sliding distance makes button selection more like a goal-crossing task than a steering task [1, 2]. Since input strokes are rarely in perfect alignment with the track, too short a track can lead to accidental activation during selection of nearby widgets, while too long a track can make button activation into a more difficult steering task.

Crossing

Sliding Buttons share many similarities with goal crossing widgets since a user's finger need not land directly on a Sliding Button in order to activate it, but must only slide over it. Accot and Zhai [2] have found that, for stylus input, people have an easier time handling motion constraints that are orthogonal to the pointing movement than they do handling collinear motion constraints. This can lead to potential reductions in selection time and errors in interfaces that use orthogonal constraints and avoid the collinear motion constraints found in pointing-based interfaces. For example, Sliding Buttons placed at window edges with an outward orientation can be thought of as orthogonally-constrained targets with practically infinite width along the line of movement. A similar effect can be achieved by locking all buttons after the first selection in a stroke. While such locking widgets may facilitate selection, the expressiveness of goal crossing interfaces is most apparent in non-locking scenarios.

Goal crossing interfaces allow for the fluid composition of multiple commands in a single stroke [4]. Figure 4(a) shows a simple example. The Sliding Button toggle switches are oriented so that the user may select all of the switches, or a contiguous block, in a single downward stroke [6]. A single stroke up will deselect the switches. This layout is useful when the interface designer expects users to select large groups of switches but makes it difficult to select just one, as only a very short stroke would avoid accidental selections of neighboring switches. A designer can facilitate selection of only one or two switches by using a layout of alternating orthogonal directions (Figure 4(b)). This layout is especially useful for tightly packed widgets. Figure 4(c) shows a compromise solution. Setting the sliding direction at 45° to the column line allows users to select multiple switches by stroking along the column or to select a single switch with an orthogonal stroke [4]. However, the visual presentation of this last example is more complex and may be difficult to interpret.

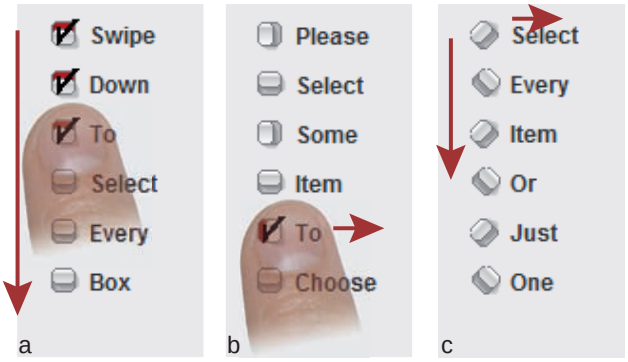

Figure 4: These toggle switches can be oriented to facilitate single stroke selection (a), or individual selection (b). Diagonal orientations allow both group and individual selection.

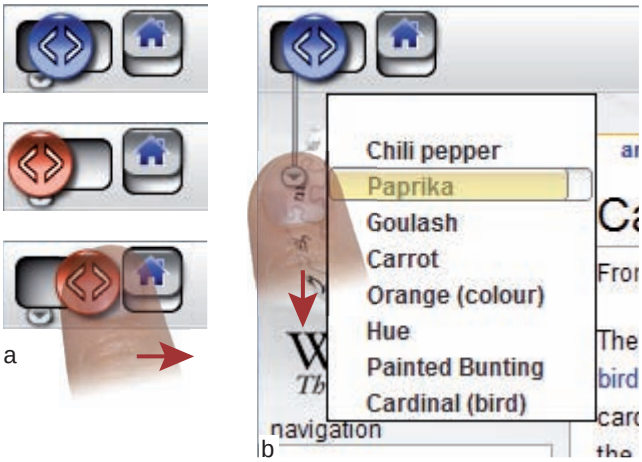

Figure 5: This multi-function browser button allows a user to move backward or forward in history with quick stroke to the left or right (a). A small pull-out knob reveals the browsing history (b).

Multi-function widgets

Sliding Widgets can load multiple meanings into the same screen area via direction multiplexing. The browser Back button in Figure 5 packs three functions into a single widget. The slider thumb's home position is at the center of the button's slot, rather than at one end. A flick to the left requests the previous page in the browsing history, while a flick to the right requests the next page. These directions are chosen in analogy to the direction of reading to provide a simple mnemonic. A small pull-out knob located under the slider thumb gives access to a drop-down list of the browsing history. Since the knob's sliding direction is orthogonal to the main slider track, users can slide it down without activating the back/forward functionality.

These widgets share not just overlapping activation footprints, as in simple Sliding Buttons, but much of the same visual footprint. This type of design is well-suited to small-screen devices where screen real estate is scarce. The compact design is not only due to direction multiplexing; the pull-out knob can be made very small thanks to area selection. Far smaller than an average fingertip, the knob would be very difficult to acquire using point selection.

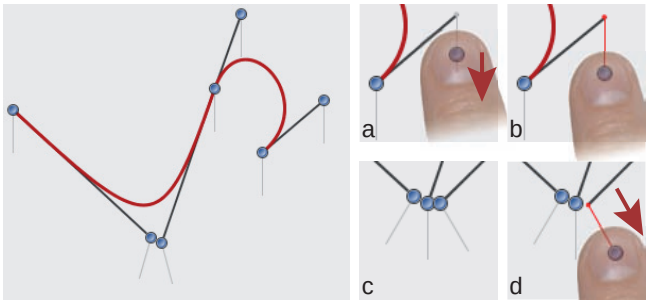

Figure 6: Area-based selection makes it easy to acquire small targets like these Control Beads. Sliding the bead down the wire (a) selects the bead, and provides an offset-cursor (b) to keep the anchor point visible. The wires of nearby beads repel each other (c) to ensure an unambiguous selection (d).

Selection Instruments

Many touchscreen selection techniques mediate input through some form of cursor in order to overcome occlusion or increase precision. Sliding Widgets require no such indirection for a large variety of interaction scenarios. However, indirection through a digital instrument may facilitate certain manipulation tasks. One example is point selection and editing in vector based illustration software. Users of such programs must frequently select very small targets, which represent curve control points, and must position them with precision. Figure 6 illustrates Control Beads, Sliding Widgets that facilitates the precise selection and positioning of curve control points. Control Beads may also be used in related tasks, such as placing point-of-interest pins on a map.

Control Beads follow a bead-on-a-wire metaphor. Before selection, the bead is centered at the control point location and is free to slide along a short dangling wire. To select a Control Bead, the user touches the bead and slides it down to the opposite end of the wire. Once there, the system produces a click sound, highlights the wire in red, and detaches the bead and wire from the point where they were anchored, allowing the use to drag them together to a new position. The wire now acts as an Offset Cursor [23]. Its anchor point, visible above the user's fingertip, allows her to position the Control Bead with no occlusion ambiguity. To release the bead, the user simply lifts her finger, and the bead springs back to the top of the wire. Note that the offset only plays a role in positioning, and not in selection, so that the finger drag state can be used for dragging screen objects rather than for tracking. Area selection makes picking an isolated Control Bead easy, but when beads are close, sliding must again be used for disambiguation. The wires of neighboring Control Beads repel each other, so that each bead slides in a different direction. The wires never point above the horizontal to ensure occlusion-free positioning. When multiple beads are near each other, the sliding directions cannot be orthogonal. As such, the selection stroke's direction vector has a nontrivial projection onto the direction vector of multiple Control Beads. The length of the wires must be sufficient to ensure unambiguous selections.

The menu selection indicator shown in Figure 7 is another example of a cursor-like instrument. The transparent Sliding Button serves to both highlight an item and then to select it with a quick flick to the right. While the item list could have been constructed simply by alternating left- and right-oriented Sliding Buttons, the presentation would have been cluttered, and half of the items would have required two separate strokes to select (one to reveal the drop-down menu, and another to select the item). The selection indicator makes for a cleaner presentation by removing the need for multiple widgets, and allows for single-stroke selection by ensuring that the selection mechanism is always at the user's fingertip. For a single stroke selection, the user pulls down the history-list knob, continues the downward stroke to highlight the desired item, and then strokes to the right to engage the slider thumb and confirm the selection. If the user wishes to pause and consider the list, she may lift her finger for a moment, and then place it back down on the selection indicator in order to slide it over another item or to confirm the selection.

The physical metaphor underlying Sliding Widgets serves only to explain their operation to the user and need not constrain their design. The design of the drop-down selection indicator illustrates several useful departures from physical constraints. The indicator does not simply move up and down at the same rate as the finger. Instead, it is gently pulled towards the nearest item so that it never straddles two items but always highlights exactly one. If the user places her finger down on a non-highlighted item, the indicator jumps to that item to ensure an unambiguous selection and a consistent activation mechanism. When the vertical component of motion exceeds the horizontal the slider thumb is locked so as to reduce the constraint on the user's movement. Likewise, when the motion is primarily horizontal the slider will not slip onto an adjacent item.

Additional Button Features

While standard pushbuttons have only two states, touched and not touched, Sliding Buttons can encode a continuum of states which allow for many useful design variations. For example, a Sliding Button can be touched but not activated. This state is analogous to the mouse tracking state and can be used for similar purposes, such as providing tool-tip help to explain a button's function. Figure 8(a) illustrates an action-preview-on-touch design [12]. When a button is touched, the system displays a preview of the action it performs. The user can remove her finger to cancel the action or slide the slider thumb to perform it. Note that the buttons must be sufficiently spaced so that only one can be touched at a time.

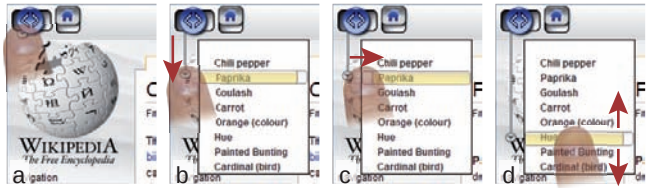

Figure 7: The selection indicator in this drop-down menu is a floating Sliding Button that plays the role of both cursor and confirmation button. It lets users select items with one L-shaped stroke, or with a sequence of strokes if they need time to read the menu.

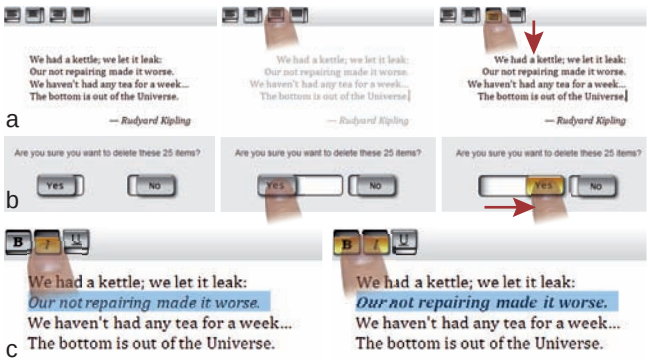

Figure 8: Sliding Buttons can use their pre-activated touch state to show a preview of the action they perform (a). Increasing the track length on contact secures the button against accidental activation when the user is distracted or mobile (b). A friction metaphor lets users press either a single button, or two at once (c).

The Slider Button in Figure 8(b) is a variation on Chu et al.'s haptic Hard-To-Press Buttons [10] that uses both the touched state and the continuous transition between the rest and activated states. The button is designed to prevent inadvertent execution of commands with potentially costly consequences (e.g. as when dismissing an accidentally activated dialog box without reading the prompt). The Yes button appears to require only a short flick to the right to activate, but the slider track extends on contact so that a short flick will fail to reach the end of the track. The user must notice the change and make a long, purposeful stroke to successfully activate the button. The long track also secures the button from activation by a finger unintentionally brushing the screen.

Figure 8(c) shows another use of the intermediate sliding state. These buttons do not use sliding as a disambiguation mechanism. Instead, they use a simple analogy to friction to determine the control to display ratio [9]. When nearly half of the finger covers the button it has a good grip on the slider thumb, with a control-display ratio of one. As the selection area covering the button decreases, the ratio drops off to zero. Thus, when the user's finger is mostly within the bounds of one button, it moves the slider thumb easily, with only a slight effect on neighboring buttons. If the user places her finger firmly between two buttons, with about half of the selection area on each, she can activate both Slider Buttons simultaneously. For this method to work effectively the buttons must be large enough to allow unambiguous selections, and simultaneous activations must have meaningful results.

EVALUATION

Touchscreen interaction based on area selection and Sliding Widgets raises many empirical questions. These include questions about touch area activation and its effects on selection and user perception of target width, and questions about widget design, such as the optimal sliding distance and the effects of multiple orientations on user motor planning. These questions are beyond the scope of this paper, but we do address one question that is important to interaction designers: do the advantages of Sliding Widgets come at the cost of selection time or accuracy? Specifically, we compare the performance of Sliding Buttons to that of pushbuttons, which are simpler to select since they require no disambiguation step. Our study does not aim to isolate any aspect of Sliding Button design or selection; it only gauges mean performance of the buttons in a typical use scenario.

Stimuli and Toolbar Interface

We model our task after the discrete selection task of Parhi et al. [21], which simulates the selection of toolbar buttons. This task accounts for the effects of adjacent targets on incorrect selections. It allows for errors due to misses, overshoots, and false interpretation of the sliding direction. We modify the task to include several targeting directions as in the work of Vogel and Baudisch [26]. This accounts for the effect of target location on user accuracy [17], and keeps the user from adjusting to a unique offset due to parallax error. Note that discrete selection tasks show different time and error characteristics than serial selection tasks such as text entry [11, 21]. Data entry is not the main target of Sliding Buttons, and our results may not extend to such tasks.

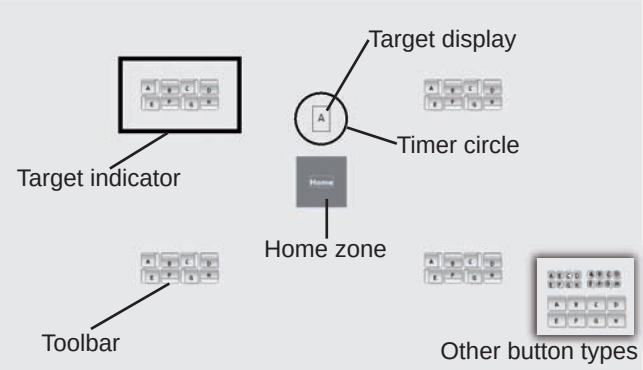

Figure 9: The experimental setup for medium size Slider Buttons. Other button conditions are shown in the inset.

The task setup is shown in Figure 9. A simulated toolbar is centered in each of the screen's four quadrants. Each toolbar is made up of eight buttons labeled A through H and arranged in two rows of four. In the middle of the screen are a square "Home" zone and a target letter display area. The center of the home zone is 71 mm from the center of each toolbar. A black outline surrounds the current target toolbar. To simulate use of a familiar interface, we chose to eliminate reading and search time from our measurements. Thus, we force the participant to keep her finger in contact with the home zone for 0.4 seconds, while the system draws a circle around the target display area, giving her the chance to read the target letter and find the target toolbar. Early removal of the finger restarts the timer.

We do not arrange Sliding Button toolbars so as to minimize overshoot errors. Doing so would require numerous direction changes which tend to confuse users. Instead, we select slider directions so they are easy to remember. Buttons on the top row slide up and to the right, while sliders on the bottom row slide down or to the left. We do not use any automated overshoot correction, as we are interested in establishing baseline performance without complex enhancements.

Unlike pushbuttons, a Sliding Button cannot be simply represented by a colored rectangle. Slider Buttons require subtle shading to help the user distinguish the slider thumb from the track. We developed the following design through a series of iterations and informal tests. We then matched this design to the visual design of the pushbuttons. The button designs used in the experiment are shown in Figure 9. The entire button area, including the margins, is active and responds to touch. There is a one pixel visual margin above and to the left of the button, and a two pixel margin below and to the right, in order to accommodate a slight drop-shadow effect. This drop-shadow is drawn only around the slider thumb. Pushbuttons are drawn as though the slider thumb fills the entire button. Small buttons are 17 by 18 pixels (3.5 by 3.5 mm) and medium buttons are 39 by 32 pixels (7.75 by 6.25 mm). These sizes were chosen in a pilot study to represent realistic difficult and easy targets. Average target size for existing hand-held touchscreen devices is around 3.8 mm [21].

Sliding Buttons were activated by sliding the thumb for 7 pixels (horizontal) or 8 pixels (vertical) in small buttons. For medium buttons these distances were 9 and 10 pixels. The selection area was a circle 28 pixels (5 mm) in diameter. It was rendered during finger contact as a subtle translucent red circle and was occasionally visible during fast flicks, due to rendering lag. Pushbuttons were activated using a take-off strategy as is done on current touchscreen devices [21]. On contact, a pushbutton is drawn in a depressed state, and its outline is drawn in red. A small cross-hair cursor indicated the selection point but was rarely visible due to obstruction by the finger. Once a button is activated, its outline is drawn in red for 0.2 seconds, and a click sound is produced. An incorrect selection (including selecting multiple Sliding Buttons) produces a bell sounds. No sound is produced when participants touch unused space surrounding buttons or touch a Sliding Button without activating it, as this would not cause an error in a real system.

Experimental Design and Procedure

A within-subject full factorial design was used. The independent variables were button type (Pushbutton and Sliding Button) and button size (Small and Medium). Testing for each of the four conditions was conducted in 4 blocks of 25 target acquisition trials per block, with the first block serving as practice. The order of presentation of the four conditions was balanced according to a Latin square. The target button and toolbar were selected randomly from a uniform distribution. In summary, the experimental design (without practice blocks) was:

	12	Participants
×	2	Button Types (Pushbutton, Sliding Button)
×	2	Button Sizes (Small, Medium)
×	3	Blocks
×	25	Trials
	3,600	Total Trials

Participants spent approximately 30 minutes performing the task, filling out a short questionnaire, and giving informal feedback on a variety of Sliding Widgets. The experimenter began by explaining the task and the operation of the two button types, and ensuring that the participant was able to select each type and size of button. Participants were asked to perform the task as quickly and accurately as possible.

To begin a trial, a participant places her finger on the home zone (outlined by the target indicator and colored green). The home zone then turns a pale blue, the target indicator shifts to surrounds the target toolbar, and the target letter is shown in the target display area. When the timer circle is complete, the participant attempts to select the target letter from the indicated toolbar. The home zone turns gray when it loses contact with the finger. Upon a correct selection, the target indicator moves back to the home zone, and the zone is again drawn in green. The participant must continue to attempt a selection until she selects the target button and no other button.

Apparatus

The experiment was performed on a Dell Latitude XT Tablet PC using a 1.33 GHz Intel Core2 Duo CPU. Touch input was provided via an N-trig capacitive touchscreen measuring 267 mm by 163 mm with a resolution of 1,280 by 800 pixels.

Participants

Twelve volunteers (four women) 24 to 50 years of age were recruited from our institution. Seven of the participants rarely used touchscreens, and five had used touchscreens from one to several times a day (ATM, ticket machine, or smart-phone). Two participants were left handed.

Results

Selection Time Selection time was measured from the time the finger left the home zone to the time the correct button was activated (either on take-off or slider reaching end of track). Trials on which an error was made, or which required more than one attempt, were eliminated from the data set, and the mean of the remaining trials was computed. Results are summarized in Figure 10(a). An analysis of variance revealed a significant main effect for Button Size, $\mathrm{F}_{1,11} = 101.05, p < 0.05$, but no significant effect for, or interaction with, Button Type. Two planned comparisons between Button Type within Button Size found no significant difference. The 95 % CI about the mean difference between Sliding Button and Pushbutton were -0.11 to 0.14 s for Small and -0.08 to 0.10 s for Medium. We expect that difference in selection time is not likely to be of practical importance to interaction designers.

Percent Correct Percent Correct is defined as the percent of trials where the participant activated the correct button, and only the correct button, in only one attempt. Note that it is possible for a participant to select the correct Sliding Button and then, in the same stroke, select another button. We count this as an error in order to keep motivation for accuracy the same as for Pushbuttons. Results are summarized in Figure 10(b). An analysis of variance revealed a significant main effect for Button Type, $\mathrm{F}_{1,11} = 8.67, p < 0.05$, and Button Size, $\mathrm{F}_{1,11} = 114.70, p < 0.05$, as well as an interaction between the two factors, $\mathrm{F}_{1,11} = 7.36, p < 0.05$. For the Small condition, using a Sliding Button yielded sig-

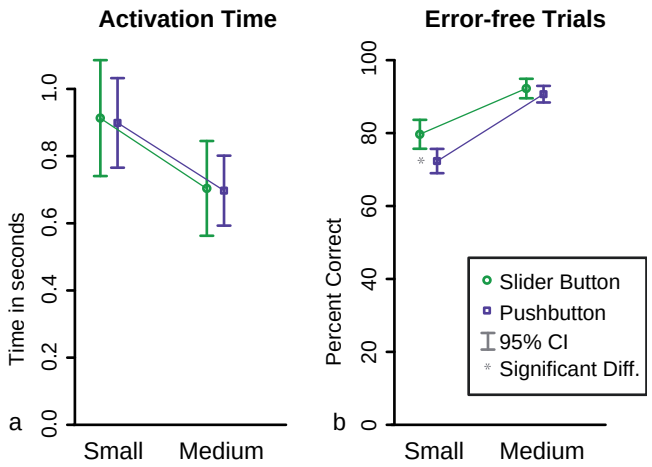

Figure 10: Activation time for perfect trials (a). Percent of trials where only the correct button was activated on the first attempt (b).

nificantly more correct selections, $t_{11} = 3.70, p < 0.025^{\dagger}$, with a mean difference of 7.33 Percent Correct between the two conditions. No significant different was found between Button Type in the Medium condition, with a 95 % CI of -2.20 to 5.31 % on the mean difference between Slider Button and Pushbutton. Thus, we conclude that Sliding Buttons can increase selection accuracy for small buttons, and have no negative effects on selection of medium size targets.

User Feedback Overall, participants had no trouble learning to use Sliding Buttons, although two had some trouble distinguishing the slider thumb from the track when first presented with Small size buttons. Common early problems included making curved strokes, which activated orthogonally placed buttons, and accidental activation due to overshoot. Users were able to quickly adjust for these errors. Most participants felt that slider buttons helped them select small targets, with 10 of 12 preferring to use Sliding Buttons for small buttons. However, participants did feel that Sliding Buttons required a bit more effort to use than Pushbuttons, with 9 of 12 stating a preference for Pushbuttons for medium buttons. Specifically, participants cited difficulty sliding their finger upwards on the screen, saying that they felt that their finger would often stick or skip when sliding up. Participants also cited the need to avoid overshoot or starting a stroke too early. Many recognized that Sliding Buttons oriented up and down were easier to select than those oriented left and right, as there were no nearby constraints on strokes in those directions. Participants responded favorably to the idea of automatic overshoot correction by blocking input after the first button was activated in the course of a finger stroke. While trouble sliding upwards and overshoot were the primary concerns raised by participants, some did mention that selecting a sliding direction seemed to require a bit of extra thought.

Following the experiment, participants were asked to try out a number of the Sliding Widget designs presented in this paper. Participants readily understood the bidirectional back/forwards button, and were able to use the pull-out knob and history menu with only a trivial amount of instruction. All participants quickly grasped how to select multiple checkboxes in a single stroke and appreciated this functionality. While most participants were able to understand how to use Control Beads without assistance, a few did not. Once prompted that "the widgets work by sliding" all were able to understand their function through experimentation. Common initial errors in using Control Beads included neglecting to activate the bead when attempting to move it upwards, and expecting the bead to drop directly under the finger instead of at the visible anchor point. Occurrence of these errors dropped off sharply after a minute or two of use. Participants showed appreciation for the ability to manipulate two sliders simultaneously and readily understood how to do so. All participants required multiple attempts (2 to 6) to resize a window using standard point selection but were able to capture the resize handle nearly every time using area selection.

[†]Bonferonni correction for two planned comparisons at $\alpha = 0.05$.

Discussion

The results of our experiments indicate that selection area activation of Sliding Widgets is a promising strategy for touchscreen interaction. We found no reduction in performance, and an improvement in accuracy for small targets. While Sliding Buttons do require more effort of the user, the larger activation area due to area selection appears to compensate for this effort.

Note that the above analysis of Percent Correct only looks at perfect trials, as there are many possible definitions of error. For example, since multiple Sliding Buttons can be selected in a single stroke, participants often activated the correct button several times before activating only that button. This simulates an application that ignores input when two commands are activated in one stroke. We chose this error model to get a baseline of Sliding Button performance without any extra features. To get an idea of performance when buttons are locked after one is activated during a stroke, we can look at the percent of trials where the first activated button is the correct one. A similar metric for Pushbuttons would also look at the percent of trials where the first activated button is the correct one and discount attempts that do not press any buttons. For small buttons, percent of trials where the first button pressed is correct are 96 % for Sliding Buttons, and 86 % for Pushbuttons (significant difference, $t_{11} = 6.12, p < 0.025^{\dagger}$). For medium buttons the values are 99 % for Sliding Buttons and 98 % for Pushbuttons (not a significant difference). While this theoretical performance seems approximately the same for medium buttons, it would be accomplished in fewer attempts for Sliding Buttons than for pushbuttons. The above analysis only considers time for perfect trials. However, had we looked at time to first correct activation we expect the results would have been skewed in favor of Sliding Buttons due to fewer required attempts. While we did not collect data for number of attempts to first correct activation, the mean number of attempts per trial for Sliding Buttons (1.30, 1.10 for small and medium buttons) were not more than for Pushbuttons (1.45, 1.11 for small and medium buttons respectively).

The theoretical performance of Sliding Buttons with overshoot correction is promising. Even without an explicit mechanism, similar performance would be expected of outward-

facing Sliding Buttons at window edges. However, Sliding Buttons do seem to require a bit more cognitive effort for planning, and may require extra physical effort for making the stroke. Tapping the screen is simpler and easier than stroking. It is important for a designer to weigh the benefits of Sliding Buttons against the additional effort they require. While much of this effort may be reduced by careful design, the expressiveness of Sliding Widgets is not always necessary, especially given sufficient screen space. However, if screen space is not an issue, we believe that selection area activation will still be beneficial. When graphical elements are sufficiently spaced, using a selection area allows for visually smaller targets, reduces parallax issues, and avoids confusion at the edges of objects. Furthermore, it allows for an interaction framework where push and sliding activated widgets can be freely mixed, allowing an interaction designer to use each type of widget as appropriate.

LIMITATIONS

There are a number of limitations on the design of interaction based on Sliding Widgets. Our informal observations, and comments from participants in our experiment, indicate that sliding direction cannot be assigned arbitrarily. While using multiple directions allows an interface to disambiguate between many items, it can confuse users. Sliding direction must be assigned in a consistent and easy to remember manner in order to provide a pleasant user experience. Arbitrary orders, meant only to optimize disambiguation, often lead to errors where users attempt to slide the widget thumb in the wrong direction. Interfaces should also avoid requiring users to slide up on the screen. While all of our users were able to slide upwards, many found it to be an uncomfortable gesture, so the direction should be used sparingly in setups similar to that in our study. Limitations on slider directions may be even greater in applications designed for one-hand thumb operation.

Another issue with sliding is that it requires a light touch and a screen with a low coefficient of friction. Some of our users found that sweat made their fingers stick. This reduced precision in tasks such as positioning Control Beads and was especially problematic when sliding the finger upwards on the screen. It is possible that Sliding Widgets are not appropriate for resistive touchscreens which require significant force to register a contact. However, since false and accidental touches are less of an issue with Sliding Widgets, it may be possibly to design these screens to require less force.

Finally, a strong limitation on Sliding Widget design is that the widget must look like a slider—it must appear to have some movable sliding element that explains its operation. These visuals require space that could instead be used for labels or icons. Push activated widget have few limitations on their visual design, as users learn their operation simply by touching them.

FUTURE WORK

We have presented a variety of novel buttons and widget designs that use sliding and selection areas. Many of these can be improved, and many more are yet to be discovered. For example, selection accuracy can be improved using various strategies for overshoot correction, such as locking and dynamic adjustment of the control-display ratio. Snapping a stroke's direction to the direction vector of the nearest widget can facilitate selection of tightly packed widgets. Rather than overloading commands on sliding direction, multi-stop Slider Buttons could use sliding distance to join related commands. Other designs can also prompt the user to draw a stroke on the screen. Dials, levers, and more complex mechanisms, may be good candidates for widget designs.

While we only explored the use of a fixed-sized circular selection area, better approximations of the contact area have many potential uses. For example, precise selections may be performed with a light touch of the little finger, and multiple selections by a broad sweep of the hand. Widgets could also take advantage of multiple points of input for parallel control of multiple parameters.

It is important to ensure that Sliding Widgets can support all functions provided by current graphical control widgets. To that end we propose investigation into Sliding Menu design. Designs may be based on the common cascading drop-down menu, as illustrated by our web browser example, or on mark based menu designs. While we expect that most legacy applications would not be trivially transformed to use a selection area instead of a point, certain instances, such as web-pages and forms, may allow for automatic widget layout and slider direction optimization.

The results of our experiment show that Sliding Buttons are a viable replacement for pushbuttons, but they do not answer many questions regarding button design. What are the speed and accuracy trade-offs of various button designs? How can designers reduce the additional cognitive effort they require? How should designers choose sliding distance and direction? Contact area selection is also worthy of study on its own. We have yet to learn how it affects user perception of target width, or how users perceive contact area size. Understanding these effects is important for selecting effective button size and spacing.

CONCLUSIONS

Standard approaches for solving touchscreen interaction difficulties either increase the size of control widgets or provide pixel-level precision via a cursor-style indirection. These approaches are a legacy of the mouse- and cursor-based interfaces that have formed the predominant model of graphical interaction in the past decades. We have shown that improving the approximation of contact by using a small selection area rather than a point is an alternative solution to selection issues on touchscreens. While using a selection area at first appears to introduce selection ambiguity, rather than to resolve it, we argue that the ability to touch multiple objects at once is an inherent property of human touch. Selection areas simply make this property apparent to interaction designers, allowing them to take better advantage of people's manipulation skills and to provide appropriate disambiguation mechanisms when necessary. Sliding is but one way of selecting one of several crowded targets, but it is an effective method that offers many of its own advantages. This new strategy addresses many existing issues in touch interaction and allows for many expressive new designs.

ACKNOWLEDGEMENTS

We thank Pierre Dragicevic, Fanny Chevalier, Jean-Daniel Fekete, and Anastasia Bezerianos for many useful suggestions and enlightening discussions.

REFERENCES

1. J. Accot and S. Zhai. Beyond fitts' law: models for trajectory-based hci tasks. In *Proceedings of CHI '97*, 295–302, New York, NY, USA, 1997. ACM.
2. J. Accot and S. Zhai. More than dotting the i's — foundations for crossing-based interfaces. In *Proceedings of CHI 2002*, 73–80, New York, NY, USA, 2002. ACM.
3. P. A. Albinsson and S. Zhai. High precision touch screen interaction. In *Proceedings of CHI 2003*, 105–112, New York, NY, USA, 2003. ACM.
4. G. Apitz and F. Guimbretière. Crossy: a crossing-based drawing application. In *Proceedings of UIST '04*, 3–12, New York, NY, USA, 2004. ACM.
5. Apple Inc. Apple iPhone. `http://www.apple.com/iphone/`.
6. P. Baudisch. Don't click, paint! Using toggle maps to manipulate sets of toggle switches. In *Proceedings of UIST '98*, 65–66, New York, NY, USA, 1998. ACM.
7. P. Baudisch and G. Chu. Back-of-device interaction allows creating very small touch devices. In *Proceedings of CHI 2009*, New York, NY, USA, 2009. ACM.
8. H. Benko, A. D. Wilson, and P. Baudisch. Precise selection techniques for multi-touch screens. In *Proceedings of CHI 2006*, New York, NY, USA, 2006. ACM.
9. X. Cao, A. D. Wilson, R. Balakrishnan, K. Hinckley, and S. Hudson. Shapetouch: Leveraging contact shape on interactive surfaces. In *Proceedings of TABLETOP 2008*. IEEE, 2008.
10. G. Chu, T. Moscovich, and R. Balakrishnan. Haptic conviction widgets. In *Proceedings of Graphics Interface 2006*, Kelowna, Canada, 2009.
11. H. A. Colle and K. J. Hiszem. Standing at a kiosk: effects of key size and spacing on touch screen numeric keypad performance and user experience. *Ergonomics*, 47(13):1406–1423, 2004.
12. C. Forlines, C. Shen, and B. Buxton. Glimpse: a novel input model for multi-level devices. In *Proceedings of CHI '05*, New York, NY, USA, 2005. ACM.
13. P. Kabbash and W. A. S. Buxton. The "prince" technique: Fitts' law and selection using area cursors. In *Proceedings of CHI '95*, 273–279, New York, NY, USA, 1995. ACM.
14. A. K. Karlson, B. B. Bederson, and J. SanGiovanni. Applens and launchtile: two designs for one-handed thumb use on small devices. In *Proceedings of CHI 2005*, 201–210, New York, NY, USA, 2005. ACM.
15. M. W. Krueger, T. Gionfriddo, and K. Hinrichsen. Videoplace: An artificial reality. In *Proceedings of CHI '85*, 35–40, New York, NY, USA, 1985. ACM.
16. G. Kurtenbach. *The Design and Evaluation of Marking Menus*. PhD thesis, University of Toronto, 1993.
17. M. Leahy and D. Hix. Effect of touch screen target location on user accuracy. In *Proceedings of The Human Factors and Ergonomics Society Annual Meeting*. ACM, 1990.
18. Microsoft. Pen and Touch Input in Windows Vista. `http://msdn.microsoft.com/en-us/library/ms702418(VS.85).aspx`.
19. A. Olwal, S. Feiner, and S. Heyman. Rubbing and tapping for precise and rapid selection on touch-screen displays. In *Proceedings of CHI 2008*, 295–304, New York, NY, USA, 2008. ACM.
20. Opera Software. Introducing Opera Fingertouch. `http://labs.opera.com/news/2009/03/05/`.
21. P. Parhi, A. K. Karlson, and B. B. Bederson. Target size study for one-handed thumb use on small touchscreen devices. In *Proceedings of MobileHCI 2006*, 203–210, New York, NY, USA, 2006. ACM.
22. C. Plaisant and D. Wallace. Touchscreen toggle design. In *Proceedings of CHI '92*, 667–668, New York, NY, USA, 1992. ACM.
23. R. L. Potter, L. J. Weldon, and B. Shneiderman. Improving the accuracy of touch screens: an experimental evaluation of three strategies. In *Proceedings of CHI '88*, 27–32, New York, NY, USA, 1988. ACM.
24. J. Rekimoto. Smartskin: An infrastructure for freehand manipulation on interactive surfaces. In *Proceedings of CHI 2002*, 113–120. ACM, 2002.
25. A. Roudaut, S. Huot, and E. Lecolinet. Taptap and magstick: improving one-handed target acquisition on small touch-screens. In *Proceedings of AVI 2008*, 146–153, New York, NY, USA, 2008. ACM.
26. D. Vogel and P. Baudisch. Shift: a technique for operating pen-based interfaces using touch. In *Proceedings of CHI 2007*, 657–666, New York, NY, USA, 2007. ACM.
27. D. Wigdor, C. Forlines, P. Baudisch, J. Barnwell, and C. Shen. Lucid touch: a see-through mobile device. In *Proceedings of UIST 2007*, 269–278, New York, NY, USA, 2007. ACM.
28. A. D. Wilson, S. Izadi, O. Hilliges, A. Garcia-Mendoza, and D. Kirk. Bringing physics to the surface. In *Proceedings of UIST 2008*, 67–76, New York, NY, USA, 2008. ACM.
29. A. Worden, N. Walker, K. Bharat, and S. Hudson. Making computers easier for older adults to use: area cursors and sticky icons. In *Proceedings of CHI '97*, 266–271, New York, NY, USA, 1997. ACM.
30. K. Yatani, K. Partridge, M. Bern, and M. W. Newman. Escape: a target selection technique using visually-cued gestures. In *Proceeding of CHI 2008*, 285–294, New York, NY, USA, 2008. ACM.