Finger-Based Target Acquisition for Hand-Held Devices
Technical Report HMT-11-03

Sean T. Hayes

Abstract

Finger-based target selection for hand-held touch screens requires new methods to be effectively and accurately achieved. Key insights for further discovery are obtained from previous work in thumb- and cursor-based interaction. Semantic Pointing, Snap-and-Go, and the Bubble Cursor are presented in a comparison review with applications to touch interaction. The strengths of thumb-based interaction techniques are summarized including TapTap, MagStick, and Escape. Elements of these cursor-based and thumb-based techniques are incorporated into the design of a novel touch-based technique, titled Drag-to-Select. Drag-to-Select inverts the traditional mental model for interaction such that moving the interface brings the target to a static cursor for selection. It is hypothesized Drag-to-Select will improve performance by reducing occlusion of both the cursor and the target; increasing the target’s effective width; reducing the distance to the target in motor space; and providing visual cues.

Index Terms

control-display ratio, Fitts’ law, hand-held device, mobile, occlusion, touch, visual cues.

I. Introduction

ONE of the commonly assumed benefits of touch input is direct interaction: Virtual objects (i.e., selectable items) can be directly touched for manipulation rather than through an external device. However, direct interaction becomes increasingly difficult as objects decrease in size. When physical objects become too small to handle, a tool (e.g., tweezers, needle, etc.) is necessary for effective manipulation. As a result, a layer of indirection is added to the interaction. Similarly, small objects on a screen require some additional design constructs to support precise interaction.

Selection difficulties for touch interaction are collectively known as the “fat-finger” problem [1]. Fingers occlude portions of the interface during interaction and have large contact areas that are measured with some uncertainty. In a traditional direct selection, the finger covers the point of contact and the desired target, reducing the feedback necessary for accuracy. Furthermore, the contact area of the finger, which is much larger than a single pixel, is the only information the interface commonly has to estimate the user’s desired single selection point. Mobile use compounds interaction challenges by reducing the accuracy of input and restricting interactions due to the highly limited surface area as compared to desktop and tabletop devices.

Researchers are exploring interaction paradigms that mitigate the fat-finger problem in order to provide efficient, easy to use, interfaces for hand-held devices that are able to accomplish tasks commonly performed on a stationary computer. A good design is simple to understand and explain; supports selection of any single display coordinate (not just predefined targets); and incorporates
visual cues of appropriate interactions. Also, objects should be selectable from any portion of the screen without significant fatigue.

This paper presents a discussion of the performance of cursor-based and thumb-based techniques. Insights to improving selection speed, accuracy, and error rates are revealed. The Drag-to-Select technique is described in Section V, which leverages key characteristics the described techniques in order to improve ease of use with hand-held touch devices.

II. CURSOR BASED INTERACTION

Indirect touch-interaction techniques can benefit from cursor movement enhancements. Disconnecting the targets and the contact points provides many possibilities that reduce occlusion. Cursors are a standard for representing indirect control of the virtual space. This section provides a description of how performance has been modeled with cursor-based interaction and a comparison of how these methods are used to improve performance.

A. Performance Modeling with Fitts’ Law

Fitts’ law (see Equation 1) is commonly used to model and evaluate interaction techniques for input devices [2]. The model predicts the time \( T \) required to select a target based on the distance to the target \( D \) and the target’s width \( W \). Constants \( a \) and \( b \) are estimated by linear reduction of experimental data. For improving interface design, efforts focus on reducing the index of difficulty \( ID \), which is the logarithmic portion of the Fitts’ law (see Equation 2). Accordingly, there are two design principles for improving the target-selection performance: enlarge the target and reduce the distance to the target.

\[
T = a + b \log_2 \left( \frac{D}{W} + 1 \right) \tag{1}
\]

\[
ID = \log_2 \left( \frac{D}{W} + 1 \right) \tag{2}
\]

Although Fitts’ law has been demonstrated with a variety of methods and hardware [3], new types of interaction often contain additional properties that may effect the fit of the model. Such techniques require experimental validation. Fitts’ law has been adapted for some additional variables present in touch interaction, but such variables are difficult to measure experimentally and have not been validated [4]. Gesture based interaction such as Escape [5] add another layer of complexity because directional movement is required after reaching the target. While Fitts’ law may not fully model touch-based target selection techniques, related design principles remain valid.

III. ADAPTIVE CONTROL-DISPLAY RATIO

The control-display ratio (i.e., CD ratio in Equation 3) defines the relationship between the distance traveled in the physical world (\( dX \)) and distance traveled by the virtual cursor (\( dx \)). Low CD ratio values (e.g., \( CD_{ratio} \leq 1 \)) allow larger physical movements to translate into smaller on-screen movements, providing greater precision. Larger CD ratio values increase the relative cursor speed, allowing targets to be reached more quickly and with less fatigue. Dynamic CD ratios can be used to improve performance [6].

\[
CD_{ratio} = \frac{dX}{dx} \tag{3}
\]
Fig. 1: One-dimensional example of adaptive CD ratio over a target (X is the display location, x is the control-device location, thick areas on the axes are a target’s location) [7]

The most common application of dynamic CD ratios is “cursor acceleration”, which is applied in Microsoft Windows XP/Vista/7 and Apple Mac OS X [8]. The CD ratio is a function of the device velocity. The greater the device velocity, the lower the CD ratio. The premise is that fast device movement implies a greater virtual distance to the target, which can be traversed with less motor space and less accuracy. Dynamically adjusting the CD ratio produces two separate movement spaces. That is, the distance from the cursor to the target (in visual space) is no longer linearly related to the required device movement (in device space). Another, perhaps more powerful method is adaptively adjusting the CD ratio based on target locations rather than the device velocity.

A. Semantic Pointing

Semantic Pointing supports the hypothesis that task difficulty was dependent on the motor space, not the visual space of targets [7]. The size of potential targets are increased in motor space by reducing the CD ratio when the cursor is over the objects and increasing the CD ratio when the cursor is over empty space. The density of objects in visual space represents the local ‘need for accuracy’ and therefore the relevance of each pixel to potential interaction (see Figure 1).

Semantic pointing was tested with one-dimensional discrete pointing tasks for direct analysis with regard to Fitts’ law [7]. A bell-shaped CD ratio function was used for targets (see Figure 1). The scales of objects in motor space were constant, double, and quadruple the visual size. For example a double scale indicates that a target is twice the motor size compared to its visual size. No distracter objects (i.e., objects not the intended as targets) were present, which would increase the motor distance to intended targets.

Increased motor scale of targets resulted in decreased movement time. The differences were more significant for targets with larger visual space ID (see Equation 2). Error rates were also reduced for larger motor scales. Participants did not notice the CD distortion.
The experimental data had better fit to Fitts’ law using motor distance instead of visual distance (according to the coefficient of simple determination and the root mean square error). This result supports the hypothesis that Fitts’ law is dependent on the motor space rather than the visual space. Therefore, visual sizes can be theoretically reduced to sizes related only to the constants of recognizability and legibility. Designers can also adjust the motor size of targets based on importance. Default options could be larger in motor space compared to related options. Similarly, targets that are more dangerous can be given a smaller motor size to reduce accidental selection.

B. Snap-and-Go

Snap-and-Go is an object alignment technique that inserts additional motor space in locations that are commonly aligned [9]. Traditional snapping is generally used to align objects by automatically positioning an object when it becomes close to the alignment point. The area within a snap radius is removed from motor space and therefore the related visual area becomes inaccessible. Snapping must be deactivated for object placement within the snap radius, but not directly on the snap location. Conversely, Snap-and-Go enlarges the motor space near the snap location, which allows for greater precision within the snap area by slowing the cursor within the snap radius. Fitts’ law predicts that interaction time will be slower because of the longer motor distance to the target, but a deactivation option is unnecessary.

Interaction in one-dimensional space assures that the cursor will cross the intended target. However, two-dimensional spaces allow the cursor to accidentally circumvent the intended target. To solve this problem, snap-and-go in two dimensions is supported by the introduction of two different motor space manipulation methods.

A plus widget, shown in Figure 2a, is used for simultaneous alignment on the x and y axes. The plus widget guides the cursor using pixels with directional ‘friction,’ dubbed “frixels”. Pixels in these areas are divided into multiple motor-space units on one axis. For example, a pixel in the horizontal bar of the widget will contain a vertical line of two or more motor space pixels. As the user moves the device in a straight line in Figure 2a, the added motor space guides the cursor over the target. A bar widget, shown in Figure 2b, is used for alignment on one axis. Extra motor space is added uniformly in both dimensions, avoiding unmerited lateral motion. Combinations of these widgets provided flexible snapping arrangements.

Snap-and-Go was evaluated in one dimension and in two dimensions without the described wid-
gets. Instead, the experimental interface design utilized horizontal and vertical lines of extra motor space that span the interface. The use of frixels is not mentioned and is instead a linear continuation of the one-dimensional Snap-and-Go, which intersected at single pixel target locations. Snap-and-Go was found to be faster than no snapping, but expectedly slower than traditional snapping. Performance was reduced as the distracter count increased. The one-dimensional experiment modeled Snap-and-Go with Fitts’ law. The data provided suggests that the visual space instead of the motor space was used for measurement. Participants found visual cues indicating locations of changing pointer acceleration (i.e., motor space manipulation) helpful.

The authors point out that alignment techniques can be utilized for target selection as well as alignment. Targets act as alignment points and the cross widget is centered on the target. Frixels may extend beyond the border of the target in all four directions. Like Semantic Pointing, both techniques increase the target motor areas to aid in selection. The implementation of Semantic Pointing evaluated by Baudisch et al. also extended the motor space beyond the target [7]. However, this extension was to avoid CD ratio discontinuities and was not presented as a benefit of the design. Semantic Pointing extends the motor space in all directions with a maximum CD ratio at the object’s center. Unlike Semantic Pointing, Snap-and-Go extends the motor space only on the axes from the center of the target using frixels, which guides the cursor to the target even when the cursor would not have originally passed over the target.

C. Bubble Cursor

Instead of point selection, area cursors select any object contained within the cursor’s area [10], [11], [12]. The Bubble Cursor is a unique design because the area of the cursor dynamically resizes to maintain that exactly one target is always within cursor’s selection area. Also, the cursor area is circular, which insures that the closest target to the cursor’s center is always the one contained.

Figure 3 demonstrates how the cursor resizes between two objects. In Figure 3a, the cursor size contains the closer object. After cursor movement to the right in Figure 3b, the larger target is closer to the cursor and is captured. If the cursor area cannot expand to cover a single object, it morphs to cover the closest object as shown in Figure 3c.
The authors point out that the Bubble Cursor essentially maximizes the size of objects in motor space. One is always covered by the cursor, which effectively expands an object area by filling all available motor space. As a result, the interaction can be modeled with Fitts’ law using the effective target width, (i.e., the maximized target area). However, the modeling is more complex due to the non-rectangular target shapes in the motor space.

The Bubble Cursor is the only described technique to test the applicability of Fitts’ law in two-dimensional space. The motor space was utilized measuring target widths and traveled distances. Two experiments that included distractors and varied the targets visual width, effective width, and distance found strong linear reduction to Fitts’ law. Interestingly, there was also a significant effect for the visual target width. Area cursors do not manipulate the CD ratio and therefore are applicable to direct touch interaction [13].

Compared to a traditional cursor, the Bubble Cursor was significantly faster for all 27 tested conditions, even with the smallest increases in an object’s effective width. The Bubble Cursor also reduced error occurrences. Unexpectedly, movement time increased as the density of objects decreased, which may be due to the visual distraction of the cursor becoming extremely large [12].

IV. Finger-Based Interaction

The following designs seek to specifically improve performance by mitigating the limitations of touch input. Two commonly compared techniques to reduce the effects of finger occlusion are Offset Cursor [14] and Shift [15]. Offset cursor moves the cursor a set distance from the contact point. Shift displays a copy of the occluded area on any side of the contact point.

A. TapTap

TapTap was specifically designed for thumb interaction on small touch-screens [16]. Two taps of the thumb perform a single selection. The first tap, placed near a target, defines an area that is subsequently magnified and centered within the screen (see Figure 4a and 4b). Target areas are magnified disproportionately more than the zoom area. The second tap can then select the enlarged targets with greatly reduced occlusion (see Figure 4c). Centering the zoom area reduces contact with the edges of the screen, which have greater sensor error.

TapTap requires two taps, but reduces movement time for each step. With the first tap, the user selects the region containing the target for magnification. For the second tap, the target size is
increased and is reposition near the center, which reduces the distance between the thumb’s resting position and the target. Therefore, Fitts’ law predicts a reduced $ID$ for both tasks.

TapTap had an insignificantly longer interaction time than direct touch, but had the lowest error rate of all interaction types evaluated including Offset Cursor [14] and Shift [15]. TapTap was also subjectively preferred over the other methods.

B. MagStick

MagStick is an example of the applicability of cursor-based techniques to touch interaction. MagStick utilizes a virtual telescopic stick with a “magnetized” end [16]. The initial contact position defines a pivot point for the stick (see Figure 5a). Dragging the finger in the opposite direction of the target (see Figure 5b) extends the the stick the same distance toward the target. Once close proximity to the target is reached, the endpoint is drawn into the target’s center using a variation of semantic pointing [7] (Figure 5c). MagStick is an indirect interaction technique that can be modeled with Fitts’ law in the manner used by semantic pointing. The reduced motor space around targets reduces the $ID$.

MagStick was evaluated in the same study as TapTap. MagStick performed more slowly than direct touch and TapTap, but was faster than other techniques including Offset Cursor and Shift. MagStick had fewer errors (10.4%) compared to direct touch (58%), but the results were not significant.

C. Escape

Escape, a thumb based target selection technique, disambiguates closely spaced objects using directional strokes [17]. A finger is placed near the target and dragged in the direction indicated by the target icon (see Figure 6). The target’s effective width is increased because direct-touch contact is not required.

Escape was significantly faster than Shift for all tested target sizes (6, 12, 18, and 24 pixels). The
TABLE I: Comparison of the touch-based techniques

<table>
<thead>
<tr>
<th></th>
<th>Occludes Target</th>
<th>High Object Proximity</th>
<th>Interaction</th>
<th>Gesture Based</th>
</tr>
</thead>
<tbody>
<tr>
<td>Escape</td>
<td>Yes</td>
<td>Reduced Efficiency</td>
<td>Direct</td>
<td>Yes</td>
</tr>
<tr>
<td>MagStick</td>
<td>No</td>
<td>Greatly Reduced Efficiency</td>
<td>Indirect</td>
<td>Yes</td>
</tr>
<tr>
<td>TapTap</td>
<td>Somewhat</td>
<td>Reduced Efficiency</td>
<td>Direct</td>
<td>No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Spacial Steps</th>
<th>Location Access</th>
<th>Point Selection</th>
<th>Tracking State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Escape</td>
<td>2 (select, stroke)</td>
<td>Reduced on Edges</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MagStick</td>
<td>1 (drag)</td>
<td>All</td>
<td>Yes (w/o Sem. Pt.)</td>
<td>Yes</td>
</tr>
<tr>
<td>TapTap</td>
<td>2 (tap, tap)</td>
<td>All</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Fig. 6: Target selection with Escape [17]

difference in the error rates were not significant. Escape was also tested to determine the effects of color coding target directions and difficulty of directions. These results were non-significant, possibly due to the small sample size (only 8 participants).

D. Summary Comparison

Table I provides an overview of the attributes of the touch techniques. MagStick is the only technique described that eliminates occlusion because of its indirect pointing mechanism. TapTap is the simplest interaction, not relying on gestures or any indirect movement. Escape requires two spatial steps, but they are both performed in one fluid movement. Escape is the only technique described that alters the visual shape of objects.
D.1 Close Target Proximity

Tight layouts of widgets are common, especially on small screens. Therefore, it is important to evaluate the performance of a technique when there is little to no spacing between targets. Semantic pointing and MagStick can reduce the motor space around objects only if space is available. Similarly, target selection based on Snap-and-Go requires a snap radius around objects in order to be effective. When using a bubble cursor the effective width is identical to the virtual width of the target if targets are touching.

TapTap can increase the overall zoom of the area selected with the first tap, which increases the target’s effective area. However, the additional zoom of targets in the enlarged area cannot be applied. MagStick improves performance by reducing occlusion for any arrangement of objects, but cannot utilize the benefits of semantic pointing as described previously. Escape is theoretically the most robust for reductions in object spacing. The effective width (and visual width) of objects can overlap because the target is disambiguated by the stroke direction.

V. Proposal of Drag-to-Select

I hypothesize that fat-finger problems can be minimized through an adjustment to our current mental-model of interface design. Within a traditional interface, objects are fixed in place. The user moves a cursor to the desired target object for selection. This relationship could be reversed such that the cursor is fixed and the interface moves to place the target under the cursor. This process, dubbed Drag-to-Select, is similar to panning or scrolling a map or graphic.

Figure 7 provides an example of the basic Drag-to-Select technique. When contact with the surface is first made, the object under the cursor is highlighted to indicate that it is aligned for selection (see Figure 7a). Dragging the finger pans the interface, which places other objects under the cursor (see Figure 7b). Tapping with a secondary finger and releasing the contact makes the selection. If desirable, the interface will pan back to its default position after selection (see Figure 7c).

Within a GUI, movement could be limited to only the active region, for example the selected panel may be panned while the other panels remain unchanged. Objects that travel outside the active region may remain visible (see Figure 7b) if within the screen area. If the visual layout of objects is unimportant, objects moving out of the region could ‘wrap-around’ to appear on the opposite side in order to leverage empty space. The wrap-around could be useful for lists of objects and other widgets, but would be misleading if applied to spatial data such as maps and charts. Situations that benefit from wrap-around may not need a return to the default position after selection. For example, a list could be setup such that the current or latest selection is always on top.
A. Features

A.1 Clutching

Clutching would be a powerful feature that allows the user to continue a selection even if the edge of the touch surface was reached before selection. Clutching occurs when the hand is repositioned mid-gesture without affecting the state of the system. For screen sizes that support multi-touch, Drag-to-Select would utilize clutching by default. Removing finger contact while dragging the interface would not change the selection state.

A.2 Cancel State

MagStick and TapTap cancel the interaction by selecting an invalid target (e.g., empty space). However, this may be difficult for tightly space objects. The authors of Escape suggest that a gesture could be canceled by returning to the start position. Drag-to-Select with clutching utilizes a similar method for this important feature. A tap with a single finger resets the interface position. Finding an open area or moving back to the start of the drag is unnecessary.

A.3 Select State

Like MagStick, a selection method may occur on the release of the touch points. For mobile devices supporting multi-touch, the tap of a second finger could indicate selection. Unique selection states can be applied using different fingers [18]. For example, the interface could be dragged with the middle finger; tapping with the index finger initiates a “left click,” and a ring-finger tap simulates a “right click”.

A.4 Cursor Placement

A user study is necessary to evaluate the ideal location for the cursor. Cursor placed in the center provides equal interaction time to select objects in any corner. Alternatively, the cursor could be placed in the upper-left corner (or upper-right for left-handed users) thus eliminating the possibility of cursor occlusion. If only one object can be selected at a time from the list of objects, the cursor may always be over the selected object. Similar to an Offset Cursor [14], the cursor could appear close to the initial contact, greatly reducing the movement time of the interface drag.

B. Benefits

The simplicity of Drag-to-Select allows for interaction with any object that has an area (e.g., widgets, map items, 1-dimensional lists, etc.) Also, the technique supports precision point selection because it does not rely on interface knowledge of potential targets. Further, it does not require gesture cues (as compared to Escape).

A unique characteristic of Drag-to-Select is the flexibility in the location and movement of the fingers during selection. Selecting a target can start by placing a finger almost anywhere on the screen. Interaction is limited if the finger is near and heading toward the edge of the screen. This limitation is common with all relative positioning surfaces (e.g., touchpads, mousepads, etc.) and is easily compensated for by repositioning the contact point. Also, the movement direction is relative only to the cursor and target. This relationship is less complicated than MagStick, whose movement is based on the location of the target, pivot point, and contact point.
C. Performance Considerations

While still possible, occlusion is greatly reduced using Drag-to-Select. It is expected that users will effortlessly avoid occlusion by intuitively placing their initial contact point outside the path between the cursor and the target. If the target or the cursor become occluded, dragging can continue after readjusting the position of the hand. A user study is necessary to determine how frequently occlusion occurs and how clutching affects performance.

Fitts’ law (see Equation 1) can be applied to Drag-to-Select in the same manner as other touch-based techniques. The effective width of the target is the width of the object being dragged to the cursor.

D. Possible Applications of Motor-Space Improvements

Combining Drag-to-Select with features of the aforementioned techniques and other established features (e.g., clutching, selection states, etc.) could be extremely beneficial. The following describes some potential combinations.

Semantic Pointing could be applied to the panning motion. When the target nears and reaches the cursor, the CD ratio would be increased to add extra motor space. These enhancements do not reduce the performance of point selection operations. Space between targets could be reduced in motor space if point selection is unnecessary. Usually, panning motions are performed such that the finger’s contact point in visual space remains constant. It may be distracting if the fingers move at a different rate than the corresponding virtual movement. Furthermore, the slower CD ratio can cause occlusion as the finger moves faster toward the cursor than the target does.

Snap-and-Go can reduce the probability of occlusion caused by Semantic Pointing. Frixels only add motor space in one dimension at a time. As a result, the target’s speed is less affected by contact with the cursor. Additionally, only adjusting the motor space on lines extending horizontal and vertical from the object’s center, changes in the CD ratio are further reduced. While the problem of the finger being faster than the target is reduced, the possibility of occlusion is still present.

Setting the default level for the CD ratio such that added motor spaces never raise the CD ratio above one (i.e., \( C/D \leq 1 \)) prevents the cursor from moving slower than the finger. Touchpads commonly found on laptops utilize CD ratios well below one, allowing a small area to control a large amount of screen surface. Reducing the CD ratio reduces the motor distance between targets and, therefore, reduces point selection between targets. However, in designs requiring only the selection of objects currently defined in the interface, low default CD ratios in combination with Semantic Pointing or Snap-and-Go could be highly effective.

The goal of Snap-and-Go is to allow objects to be placed near snap locations and provide guidance for alignment. However, if the point selection outside targets is not necessary, traditional snapping could be employed. Essentially this redefines the motor space around targets as belonging to the targets, increasing their effective width. Snapping could also be employed only in the visual space; when the cursor is first over an object, the interface “snaps” so that the target is centered under the cursor. The same motor distance would be traversed to leave the target, but it is visually evident that the object is ready for selection. Snapping within an object’s borders may be useful for tightly placed objects (e.g., lists and tables). For objects of similar size and shape, snapping is similar to scroll steps that are traditionally performed with a mouse wheel or arrow keys.

MadStick avoids occlusion because movements are always away from the target. Dragging could be inverted so that the finger moves in the opposite direction. Similar to panning the interface’s field of view (FOV), moving the FOV to the right causes the interface to move relatively to the left (see Figure 8). The benefit is that a CD ratio greater than one could be applied without causing...
occlusion. Although improved performance has been demonstrated on indirect devices [19], inverted dragging may confuse users. A helpful strategy for finger movement in this case may be to drag from the cursor location to the target’s original position. This strategy prevents occlusion during the final movements of the target selection, and therefore may be efficient. As with any strategy, inverted dragging will have an additional learning overhead and is counter-intuitive. MagStick benefits from a visible control analogy (i.e., the telescopic stick). Unless a simple visual analogy can be applied to the inverted drag motion, poor performance is expected.

Drag-to-Select can benefit from static Bubble Cursor. Figure 9 shows the intermediate step for Bubble Cursor interaction applied with the same finger motion as Figure 7. Although the cursor is constantly changing size during the interface pan, the CD ratio remains constant and visually consistent with a finger’s contact point. However, for point selection within objects, the Bubble Cursor must be disabled. The Bubble Cursor could be combined with other motor space manipulation with Semantic Pointing or Snap-and-Go to further optimize the motor space. Therefore, the Bubble Cursor may prove viable for use in Drag-to-Select.

**E. Visual Enhancements**

A key feature of Escape is the directional cues indicated require stroke directions for objects. Drag-to-Select objects could indicate the dragging direction for their selection by pointing toward the cursor (see Figure 10). Arrow directions would be updated as the position of the arrow relative
Fig. 9: Drag-to-Select with a Bubble Cursor (in Gray)

Fig. 10: Drag-to-Select with Escape style icons

the cursor changes and removed when an object is under the cursor. Pointing toward the cursor allows an arrow to be quickly referenced from wherever the user’s current focus. Also, arrows reinforce the direction necessary for dragging the interface. The layout and context would have to support altering the shape of objects.

F. Limitations

As with any user interface, the precision of interaction is limited by the interface’s sensor resolution and the user’s motor control. A common approach to increasing precision, as needed, is to allow user scaling of the input. Zooming is a common task performed with panning. A two-fingered gesture can be added to Drag-to-Select to manipulate the interface scale while simultaneously panning the
target toward the cursor. In this case, the distance between the fingers controls the scale while the change in position of the first finger controls the drag motion. The cursor position remains unaffected by the zoom level. Users may find this feature additionally helpful when interacting with spatial information stretching beyond the visible portion of the screen.

Drag-to-Select interferes with non-selection type tasks such as sketching. For example a graphic artist may wish to freehand draw the border of a new object. Such interactions are not supported because any movement along the surface of the screen is interpreted as an interface drag. However, additional interaction could be supported by entering a new interface mode. For example, a two-finger tap could toggle direct target selection and sketching.

Special consideration is also necessary for object dragging. After an object is selected, the user may wish to move it. Again the user could transition to the alternate mode and use two-fingers to drag the object. As in Drag-to-Select, the object could be moved by dragging anywhere on the touch surface. While these techniques are possible through a mode shift, the switch adds interaction complexity and may confuse the user. The current mode would need to be clearly indicated on the interface.

VI. Conclusion

For accurate point and small-target selection, an indirect method is required. Cursors are the standard for visualizing indirect selection tasks. Methods to improve cursor-based interaction, such as Semantic Pointing, Snap-and-Go, and the Bubble Cursor, can be applied to touch-based interaction techniques such as Drag-to-Select. The benefits of each technique have been considered in the context of Drag-to-Select, but a user evaluation is required to quantify each method’s effectiveness. The Bubble Cursor could provide the best performance improvement without increasing occlusion. The attributes of several thumb-based techniques (i.e., TapTap, MagStick, and Escape) were also considered as extensions to Drag-to-Select.

It is believed that Drag-to-Select will improve performance over other interaction techniques by reducing errors and target selection time. Also, extensions to the techniques such as an enhanced motor space, added support of multiple selection states, and visual cues may prove beneficial. In the future, the author plans to further design, implement, and evaluate the Drag-to-Select technique.

REFERENCES


