LinearDragger: a Linear Selector for One-finger Target Acquisition

Oscar Kin-Chung Au
City University of Hong Kong
Hong Kong
kincau@cityu.edu.hk

Xiaojun Su
City University of Hong Kong
Hong Kong
xiaojun.su88@gmail.com

Rynson W.H. Lau
City University of Hong Kong
Hong Kong
rynson.lau@cityu.edu.hk

ABSTRACT
Touch input is increasingly popular nowadays, especially for mobile devices such as smartphones and tablet computers. However, the human finger has considerably large fingertip size and finger input is imprecise. As such, acquiring small targets on a touch screen is still a challenging task. In this paper, we present the *LinearDragger*, a new and integrated one-finger target acquisition technique for small and clustered targets. The proposed method has three advantages. First, it allows users to select targets in dense clustered groups easily with a single touch-drag-release operation. Second, it maps the 2D selection problem into a more precise 1D selection problem, which is independent of the target distribution. Third, it avoids finger occlusion and does not create visual distraction. As a result, it is particularly suitable for applications with dense targets and rich visual elements. Results of our controlled experiments show that when selecting small targets, *LinearDragger* takes about 70% and 30% less selection time than target acquisition without using any techniques and with the state-of-the-art target acquisition technique that involves a single touch operation, respectively, while maintaining a reasonable error rate.

Author Keywords
target acquisition; touch input; dense target selection.

ACM Classification Keywords
H.5.2 User Interfaces: Graphical user interfaces (GUI), Interaction styles; I.3.6 Methodology and Techniques: Interaction techniques

General Terms
Human Factors; Design

INTRODUCTION
Touch input is increasingly popular in the past decade due to the popularity of mobile devices, such as smartphones and tablet computers. Although it is extremely convenient and intuitive to use our finger, instead of a mouse, as an input device, it has one limitation. Human fingers have considerably large finger tip size, referred to as the "fat finger" problem [4], making it a challenging task for users to acquire small targets from dense target clusters due to two problems. First, the fat finger tip has low input precision. Second, it causes occlusion during target acquisition.

Several techniques have been introduced to address the input precision and occlusion problems in acquiring small targets. The snap-to-target technique [9, 18] partitions the entire screen space into tiles of selection regions using Voronoi tessellation, to ease small target acquisition. Although this method has been proven helpful in a sparse target layout, it shows little effect when the targets are clustered as there is no space to extend. Unfortunately, clustering of targets appears in many applications, such as selectable items in an online map (Figure 1(a)), clustered UI elements of a text editor (Figure 1(b)), crowds of characters in an online game (Figure 1(c)), and words and characters of document being selectable targets in a text editor (Figure 1(d)). To facilitate acquisition of clustered targets, *Starburst* [2] partitions the screen space by expanding clustered targets to form selectable regions. However, this often produces long and thin selectable regions, which may be difficult to acquire by the user. *Escape* [25] adopts both the contact positions and swiping directions to disambiguate the selection of targets in dense clusters. However both approaches require auxiliary visual elements as visual cues for target selection, leading to undesired visual distraction and potentially degraded performance in extremely dense target distributions (e.g., characters in a text document).

Several other techniques address the occlusion problem either by displaying the occluded local region or controlling the virtual cursor with different mapping methods. *Shift* [23] displays the local region occluded by the fingertip in a nearby empty space. The user may perform a precise control in the offset display to select small targets within the local region. Several virtual cursor techniques [20, 22, 11, 21] map the movement of the contact point to the virtual cursor motion, such that the user may control the cursor and select the desired target without occluding the target region. However all these techniques still require precise 2D touch control when working with small and clustered targets.

In this paper, we present the *LinearDragger*, an integrated one-finger target acquisition technique for small and clustered targets. Besides using only the contact point to determine the selected target, *LinearDragger* allows the user to select a target in a dense clustered group easily with a single touch-drag-release operation without the need to accurately point to the

CHI 2014, April 26–May 1, 2014, Toronto, Ontario, Canada.

http://dx.doi.org/10.1145/2556288.2557096

Figure 1. Non-uniform target distributions are common in many applications: (a) map application showing restaurants and stations, (b) menus and tool icons in a text editor, (c) crowds of characters in an online game, and (d) words and characters of a document in a text editor.

desired target. If the first contact point can unambiguously identify a single target, the user just lifts the contact finger to select the target, i.e., select by tapping. However if there are multiple items within the local region around the contact point, the user could simple drag the finger to explore and select the desired target as shown in Figure 2. All potential targets within the local region are ordered according to the finger moving direction, and are associated with effective regions of equal width. This converts the 2D selection problem into a 1D one, removing the need for precise 2D touch control. Hence, *LinearDragger* is independent of target distribution. In addition, as it does not have finger occlusion and introduces no auxiliary visual elements, it causes no visual distraction to the user. Thus, the proposed method is particularly suitable for applications with dense targets and rich visual elements. We have conducted a user study to evaluate the performance of *LinearDragger* under different conditions, and found that when selecting small targets, *LinearDragger* performs faster than *Escape* and *Shift* while having similar error rates.

RELATED WORK

Target Expansion

Target expansion is a common approach to facilitate small target acquisition with a pointing device. Methods of this approach include [24, 5], which adopt dynamic control-display ratio; [9, 15, 6], which increase the target's effective width by applying an area cursor; and [10, 19], which are based on zooming in the visual space.

On a touch screen, target expansion techniques [18, 2] enlarge the effective sizes of small targets by partitioning the screen space into tiles with each tile corresponding to a single target. The user may click anywhere inside a tile to select a target. In [18], Voronoi tessellation is used for screen partitioning such that the nearest target from the contact point is always selected. However, the performance of this approach is highly dependent on the distribution of the targets, with small tiles for clustered targets. This makes it difficult to acquire small targets. *Starburst* [2] partitions the screen space by extending the tiles of all clustered targets from the cluster to the surrounding empty space, resulting in thin and lengthy pathways. Hence, the selectable regions of *Starburst* have to be explicitly specified and the user has to visually trace the region of the target. This may not be an easy task when many targets are clustered in a tight area. In addition, the display of the selectable regions, drawn as colored overlay or region boundaries, introduces undesired visual distractions.

Zooming is another common approach in expanding the target sizes for easier exploration and selection. In [3], a precise pointing technique with two-handed interaction is proposed. With a secondary finger to help adjust the level of magnification, the primary finger may perform precise selection in the zoomed region. Similarly, in [16], a bimanual zooming method is proposed, with zooming being operated by tapping with a secondary finger. Recently, a multitouch technique called *FingerGlass* is introduced [12], in which the user defines a local region with the first two contact fingers and the magnified local region is popped up for precise selection using the second hand. Although all these methods provide intuitive interaction, two-handed operation is not practical in many situations, such as working with a handheld device.

There are also zooming techniques that involve only single-finger interactions. In [16], a rubbing gesture is introduced for zooming, which can be integrated with the panning operation using only a single finger. In [21], a two-stage selection method is proposed. The user first taps on the screen to popup a magnified viewport of the local area, and then taps again to select the target within this viewport. *Excentric Labeling* [7] is a technique to label a neighborhood of objects around the cursor, in a layout comfortable for human eyes. It can be transformed to a target acquisition technique using hierarchical operations. The main drawback of these techniques is that zooming and target selection have to be performed in separated steps. In constrast, *LinearDragger* integrates zooming and target selection into a single one-finger operation, which reduces operation complexity and is applicable in almost all selection scenarios.

Occlusion Avoidance

Occlusion is a common problem of touch screen interactions – due to the size of the finger tip, selectable objects can be easily occluded by the contact finger during target selection. The *Offset Cursor* technique [20] is one of the earlier notable works to address this problem. It places a displaced cursor above the actual contact point, and the item under the cursor is selected when the user releases the finger. In [22], a stabilization technique is proposed for the *Offset Cursor* to provide faster and more accurate selection of small targets. However, the main drawbacks of *Offset Cursor* are that the precise cursor position is unknown until the finger touches the screen and it is not possible to select targets at the bottom of the screen since the offset position is always above the finger. In [23], an improved technique, called *Shift*, is introduced. It displays a copy of the local region occluded by the finger at a nearby position. The selection hotspot remains under the fingertip. The user may select any small object under the finger while watching the offset display. Results of the user study show that *Shift* is more accurate than traditional

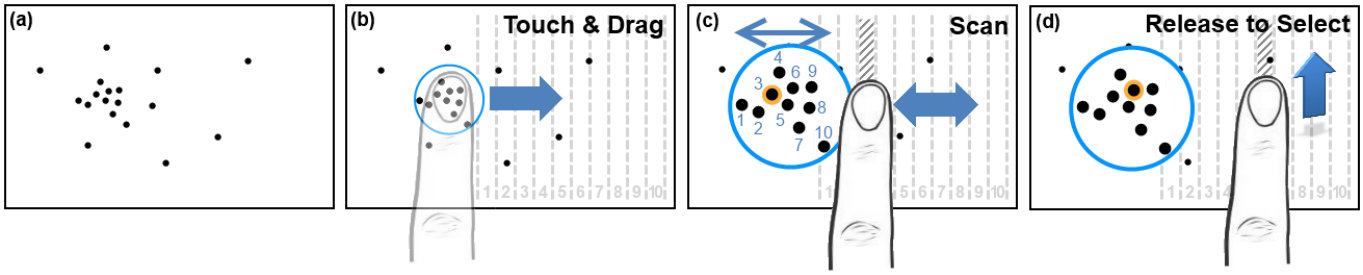

Figure 2. Overview of LinearDragger. (a) It is difficult to select a target from a clustered group of selectable objects. (b) LinearDragger is activated when the finger touches and drags on the touch screen. The contact position defines a region of interest (ROI), containing the user's desired target. (c) As the user continues dragging his finger, LinearDragger scans the potential targets within the ROI one by one. The scanning order is determined by the initial dragging direction. (d) The user just lifts the finger to select the desired target.

finger selection on small targets and faster than the original *Offset Cursor* on larger targets.

There are other virtual cursor techniques that involve complex mapping between finger gesture and cursor motion. In [3], an offset cursor located at the midpoint of two contact fingers is proposed. In [21], an occlusion avoidance method called *MagStick* is proposed. The user presses an arbitrary position on the screen to define a reference point and a virtual cursor. The virtual cursor will then be controlled by the dragging motion of the contact finger, such that the cursor motion is opposite to the dragging motion with respect to the reference point. In [18], a snap-to-target technique is applied to the virtual cursor to increase the target sizes in motion space. *Thumbspace* [11] is a technique to allow the user to reach a large screen space from a small input area using the thumb. It maps the input area to the whole screen and magnifies the user's finger motion in the input area to the mapped screen. Although *ThumbSpace* and *MagStick* resolve the occlusion problem, similar to the *Offset Cursor*, they require precise 2D touch control to acquire small targets.

Precision Refinement

Another problem of touch screen interactions is that the human finger has limited precision due to the fat finger tip. In order to acquire dense and small targets, precision refinement methods are proposed to ease target selection. In [1], *Cross-Lever* and *Precision-Handle* are introduced to increase the motion space of the virtual cursor on the touch screen. *Cross-Lever* involves controlling the intersection point between two crossed lines. It provides high precision control but is time consuming to use. *Precision-Handle* involves mapping a faction of the contact finger motion to the virtual cursor, thus enlarging the motor space and increasing the selection precision. Both techniques require separate operation steps for region selection, precise handle control and selection validation, which could potentially degrade selection performance and user experience.

In [25], a precision refinement technique called *Escape* is proposed. It allows user to select targets by swiping gestures, cued by both the target position and appearance. By swiping the fingertip in one of the predefined directions, the closest object specified with this swiping direction will be selected. In [14], the swiping gesture is applied to touch screen widgets to resolve ambiguity by assigning different swiping directions to adjacent widgets. The *Enhanced Area Cursor* [8] allows the user to first specify a coarse area and then selects the desired target within the area by invoking an angular menu with swiping gesture. However as these techniques have limited possible swiping orientations, ambiguity in swiping directions may not be completely avoided in cases of extreme dense target distributions. The swiping directions are further limited for targets near to the boundary of the screen, as the swiping gesture cannot go beyond the edge of the screen. To indicate the swiping directions of all selectable objects, a special color scheme and icon design are introduced. However, this leads to visual distractions and limits the usage of customized symbols and labels, such as the map application shown in Figure 1(a). SQUAD [13] is a progressive refinement technique for target acquisition in a 3D space. It resolves the problems of dense targets and occlusion in 3D target selection by applying a sequence of refinement operations to filter out the unwanted targets.

DESIGN AND IMPLEMENTATION

In this section, we first give an overview design of *LinearDragger* and then the implementation details.

LinearDragger is activated by dragging the finger away in any direction from the user's specified initial region (see Figure 2 and the accompanying video). A *region of interest* (ROI) is defined as the user's finger first touches the screen. We set the ROI as a circular area, with the center being the first contact point of the finger. The radius of the ROI is adjustable and its optimized value depends on the screen size and the average size of the finger tips of the user group. All selectable targets within the ROI are considered as the set of *potential targets*. We assume that one of these potential targets is the user's *desired target*.

The user drags the finger to linearly scan (and highlight) the potential targets one by one (Figure 2(b)). The order of scanning is determined by the moving direction of the contact finger. It starts from the opposite end and scans along the direction of the finger motion, as the user keeps dragging the finger away from the first contact point (Figure 2(c)). The user may select an highlighted target anytime by lifting the contact finger up (Figure 2(d)). Zooming of the ROI is invoked as the finger drags and leaves the ROI. This facilitates better visual feedback with small targets. To minimize visual

distraction and avoid unnecessary focus switching, all the visual elements about the activation areas in Figures 2, 3 and 4 (i.e., the dashed lines, the shaded region and numbers) are for illustration purpose only; they are not shown on the screen.

The distance between the current position and the first contact point of the contact finger determines which of the potential targets is focused and highlighted. This essentially defines a uniform 1D mapping between the ordered list of potential targets and the moving distance of the contact finger from the first contact point. The user may intuitively scan through the potential targets one by one by dragging the contact finger in the scanning orientation. This 1D mapping provides a *constant effective width* for all potential targets and avoids the need for precise selection in a 2D space, which is required by methods such as *MagStick* [21] and *ThumbSpace* [11].

The user is free to move the contact finger in any direction to activate *LinearDragger*. The moving direction may depend on the user's preference or the location of the ROI on the screen. This allows the user to select any targets on the screen, including those located near to the boundary/corner of the screen (Figure 3(b)).

In general, *LinearDragger* has the following nice properties:

- **Single-handed one-finger interaction** – *LinearDragger* only requires one finger to operate, which is perfectly suitable for use in a small mobile device being held by one hand and operated with a single finger.
- **Single action** – It combines activation, selection and dismissal into a single touch action.
- **Intuitive to use** – By remapping the potential targets into a linear list, it provides a simple 1D selection paradigm.
- **Insensitive to target distributions** – The targets' effective widths are independent on target density and distribution. The user can easily acquire small and clustered targets without precise operation.
- **No finger occlusion** – The contact finger always drags away from the ROI, and hence no finger occlusion occurs.
- **No visual distraction** – It does not introduce any visual distraction as no extra widgets / elements are shown. (Note that the zoomed ROI may possibly lead to visual distraction, but this zooming feature is optional and does not affect the selection operation of *LinearDragger*.)
- **Adaptable to different screen sizes** – *LinearDragger* can be operated with any finger and different device holding gestures. Thus, it is applicable to large tablets or even wall-sized touch screens.

Cancellation Mechanisms

We provide two different mechanisms for the user to dismiss *LinearDragger* and cancel the selection. The first mechanism adopts an additional waiting step after the user has selected a potential target by lifting the contact finger. Within the waiting period, the user may tap the screen at any location to cancel the selection and dismiss the focusing ROI. In our study, we set the additional waiting period as 1 second, which is long enough for the user to confirm the selected target or to cancel the selection if necessary. However, these additional waiting period and tapping step introduce extra user interaction and longer operation time.

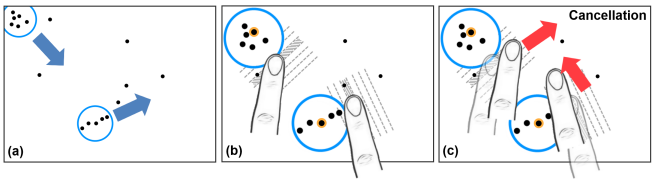

Figure 3. (a) The user may drag the finger in any direction to activate *LinearDragger*. (b) The zoomed ROI will be located within the screen, even if the original ROI may be located near to the screen boundary. (c) The user may cancel the selection and dismiss *LinearDragger* by moving the finger in the orthogonal direction to the original dragging direction.

To achieve a better performance, we have designed another cancellation mechanism, which only requires a simple step to cancel the selection. After *LinearDragger* is activated, the user may cancel the selection and dismiss *LinearDragger* by moving the contact finger roughly in the *orthogonal direction* to the original dragging direction (Figure 3(c)). This design retains the feature of a simple touch-drag-release interaction.

We have conducted a preliminary user study on both cancellation mechanisms and found that the orthogonal dragging method gives better performance.

Integration with Target Expansion Techniques

LinearDragger can be easily integrated to existing target expansion techniques. In our experiments, we have integrated *LinearDragger* with the snap-to-target approach of the Bubble cursor [9], which continuously sets the selection focus to the target closest to the cursor location. This approach effectively partitions the screen space using Voronoi tessellation.

In our implementation, we adopt the Bubble Cursor approach such that for sparse regions, the user may directly select a target by tapping on it or the nearby space without ambiguity. For dense regions, the user may activate *LinearDragger* to help select a target by a simple dragging operation.

Implementation

LinearDragger maps the 1D movement of the contact finger to the focus switching of the potential targets. This is the key for the user to efficiently access all potential targets in the ROI with intuitive and controllable input. We have experimented two different mapping functions in our implementation. The first one uses a constant effective width, as described before. This is based on the assumption that all potential targets are equally important. The second one uses a linearly scaled effective width. It is based on the assumption that a potential target nearer to the first contact point is more likely to be the intended target.

The first mapping function uses a constant effective width, as shown in Figures 2 and 3. It allows easy control in target

scanning. Specifically, it is formulated as:

$$k = \begin{cases} \lfloor (d - d_{min})/EW_{base} \rfloor + 1, & \text{if } d > d_{min} \\ 1, & \text{otherwise} \end{cases} \quad (1)$$

where k is the 1-based index of the current focused target. d is the projected distance from the current contact point to the first contact point. d_{min} is the minimal distance threshold. EW_{base} is the predefined effective width (see Figure 4(a)). The projected distance is defined as $d = (p - p_0) \cdot v$, where p and p_0 are the positions of the current contact point and the first contact point, respectively, and v is the normalized vector of the initial moving direction computed from the first few finger movement samples. The threshold distance, d_{min}, controls how long the finger should be dragged before the scanning is started, and depends on the size of the finger since the contact finger should not occlude the ROI during scanning and selection. In our experiments, we set $d_{min} = 100$ pixels (about $13.5mm$) for the main testing device, as this size is close to the average size of the finger tip.

The constant effective width gives a more predictable control. However, users may sometimes prefer the targets closer to the first contact point to be selected easier than others in the ROI. Hence, we also introduce the linearly scaled effective width, which associates different effective widths to different potential targets, with targets closer to the first contact point having larger effective widths, i.e., easier to be selected. Specifically, an extra expansion is added to the effective width of each target, such that the closest target received the maximum expansion of EW_{ex}, and the expansion reduces linearly along the two sides of the ordered list (Figure 4(b)).

We have conducted an informal preliminary study to evaluate how these two mapping methods affect the performance of *LinearDragger*. We have found that the constant effective width always gives better performance and all participants prefer this mapping method because of its predictable control. Since we hide the activation areas with no visualization support, users found it difficult to perform selection and were confused by the non-uniform size of activation areas with the linearly scaled mapping method.

To facilitate better visual feedback, a local expansion in the visual space of the ROI is optionally provided (Figures 2 and 3). The ratio of zooming is controlled by distance d (see Eq. 1), such that the magnified region will occupy the empty space between the current finger location and the original ROI. Specifically, the zooming ratio r is defined as:

$$\tilde{r} = \begin{cases} (d - d_{min})/s + 1, & \text{if } d > d_{min} \\ 1, & \text{otherwise} \end{cases} \quad (2)$$

$$r = \min(\tilde{r}, r_{max}) \quad (3)$$

where s determines the speed of zooming. r_{max} is the maximum zooming ratio. In our experiments, we set $r_{max} = 2.5$ and $s = d_{min}$.

If the orthogonal dragging method is adopted for cancellation of the selection, we need to determine when the user moves the contact finger in the orthogonal direction to the original dragging direction. We compare the magnitude of the orthogonal component of the movement vector with the pre-specified threshold: $|(p - p_0) \times v| > \tau$. In all our experiments, we set $\tau = 50$ pixels (or $6.75mm$).

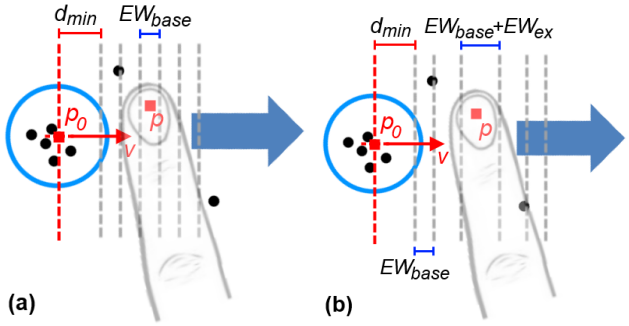

Figure 4. We have designed two mapping functions for mapping the finger motion to the scanning speed of the potential targets: (a) linear mapping with constant effective width, and (b) non-linear mapping that gives larger effective widths to targets closer to the initial contact point.

PRELIMINARY STUDY 1: SIZE OF THE ROI

Before comparing *LinearDragger* with other touch-based target acquisition techniques, we first conducted a preliminary study to evaluate how the size of the ROI affects the performance of the proposed method.

Apparatus

The experiment was conducted on an Acer W3-810 tablet computer, with an $8.1''$ ($20.6cm$) touch screen running MS Windows 8. The screen is of $17.5cm \times 11cm$ and has a resolution of 1280×800 (73.6 pixels/cm). Participants were free to hold the tablet in any way or to place it on a table. The experimental application was implemented using C# and .Net Framework, and displayed in full screen mode.

Participants

Eight participants (4 males and 4 females) between the age of 24 to 29 were recruited. They were mainly postgraduate students. All had experience with computers, touchscreen tablets and smart phones, and happened to be right handed. All participants were recruited from a university (recruitment posters were posted for open recruitment). Each of them was given a gift voucher of USD13 for participating in the user study.

Procedure and Design

At the beginning of the task, a button of $2cm \times 2cm$ was displayed at the middle-right of the screen. Once the participant tapped the button, 12 circular candidate targets of 12 pixels ($1.6mm$) in diameter were distributed randomly within a circular region of 100 pixels ($13.5mm$) in diameter, such that no targets overlap each other. In each task, participants were asked to select the desired target among the candidate targets as accurate and fast as possible. The desired target is highlighted in blue, while other candidate targets are filled in black color. Besides having the clustered targets within the circular region, 20 distracter targets of the same size as the clustered targets were placed randomly in the remaining empty space (Figure 5(a)). Each participant performed

the experiment in two separate sessions. One session was divided into two ROI size configurations: 26 and 40 pixels (or $3.6mm$ and $5.4mm$) in diameter. Another session was divided into three ROI size configurations: 52, 66 and 80 pixels (or $7.2mm$, $9.0mm$ and $10.8mm$) in diameter. For each ROI size configuration, each participant was asked to perform fifty selection tasks. Each participant spent approximately one hour in total on the two sessions.

The sizes of the ROI were chosen based on the finding by [17] that objects should be at least $9.2mm \times 9.2mm$ in size in order to keep the selection error rates low. Hence, we adopted a ROI with a similar reference size ($9mm$ in diameter), plus a larger size and three smaller sizes (40% to 120% of the reference) in this experiment.

We recorded the duration between the user tapping the start button and when the target selection task was completed. The participants were allowed to use the orthogonal dragging mechanism to cancel a selection process. If a participant selected a wrong target or cancelled the selection before selecting the desired target, the task would be restarted until the correct target was selected and the recorded time would be accumulated. Before the experiment, we explained the details of the procedure and gave the participant a short duration of 5 to 10 minutes to get familiar with the interface. A total of $8 \times 5 \times 50 = 2,000$ selection tasks were performed in this experiment, with each participant completed 250 tasks.

Results

When the size of the ROI is set to 26, 40, 52, 66, and 80 pixels ($3.6/5.4/7.2/9.0/10.8mm$), the mean selection times were $3180ms$, $2740ms$, $2280ms$, $3230ms$, and $3520ms$, and the corresponding error rates were 12.1%, 9.3%, 4.6%, 11.3% and 13.7%, respectively. There was a significant effect on the mean selection time ($F_{(4,28)}$=$25.32, p$<0.001) and the error rate ($F_{(4,28)}$=$5.71, p$<0.001).

Discussion

We observe that both the selection time and the error rate are at the minimum when the size of the ROI is 52 pixels. This may due to the fact that a larger ROI size leads to more targets being included in the scanning. Thus, participants needed to look through more candidate targets, resulting in a longer selection time and a larger error rate. On the other hand, if the size of the ROI is too small, it is much harder for the participants to specify the ROI so as to include the desired target. Hence, the participants needed to redo the selection, which affected the overall performance. As such, we fixed the size of the ROI to 52 pixels ($7.2mm$) for the remaining experiments presented in the paper, unless specified otherwise.

PRELIMINARY STUDY 2: CANCELLATION MECHANISM

We also conducted another preliminary study to evaluate the two proposed cancellation mechanisms. Note that the diameter of the ROI was set to 52 pixels ($7.2mm$) as stated in preliminary study 1.

Apparatus and Participants

Same as those in preliminary study 1.

Procedure and Design

Participants were asked to perform two different types of tasks in this experiment.

In the first session, we evaluated the selection performance of both cancellation mechanisms. We used the same selection task as described in preliminary study 1, and asked participants to select the desired targets using the specified cancellation mechanisms. Recall that with the orthogonal dragging mechanism, only a single touch-drag-release operation is used for selection, while for the other mechanism, an additional waiting time is needed to validate the selection. Each participant performed two groups of 50 selection tasks in one session, each group using one cancellation mechanism.

The second session of the experiment is to evaluate the accuracy of the cancellation mechanisms. With the same setting as the first session, we asked the participants to cancel the selection with the specified cancellation method after focusing on the desired target. (Hence, the task was completed if the desired target was the last highlighted target before the cancellation was performed.) The whole experiment lasted for about an hour and as in preliminary study 1, after the detailed explanation provided in each session, participants could practice for a short duration. In this experiment, a total of $8 \times 2 \times 50 = 800$ selection tasks and $8 \times 2 \times 50 = 800$ cancellation tasks were performed. The completion times of all selection tasks and successful rates of the cancellation tasks were recorded.

Results

Experimental results show that the mean selection time using Orthogonal Dragging was $2310ms$ with a cancellation accuracy of 97.5%, while the mean selection time using Additional Tap was $3400ms$ with a cancellation accuracy of 99.4%. Repeated measures analysis of variance shows that different cancellation mechanisms have significant effect on selection time ($F_{(1,7)} = 15.30, p < 0.001$)) but no significant effect on cancellation accuracy.

Discussion

We can see that using Additional Tap was slower than using Orthogonal Dragging by around $1s$ for the target selecting tasks. This was mainly caused by the extra waiting time needed for the system to validate the target selection. Because of their similar accuracy but different performances, we chose to use Orthogonal Dragging as the cancellation mechanism in *LinearDragger* in the main experiment.

MAIN EXPERIMENT: PERFORMANCE COMPARISON

After determining the size of the ROI and the cancellation mechanism, we conducted the main experiment to quantitatively evaluate the performance of *LinearDragger*. We compared it with unaided touch pointing *DirectTouch*, which served as a baseline, the *Bubble* Cursor [9], which is a common target expansion approach for selecting small targets, *Shift* [23] and *Escape* [25], which are two alternative single-touch target selection techniques supporting high precision selection.

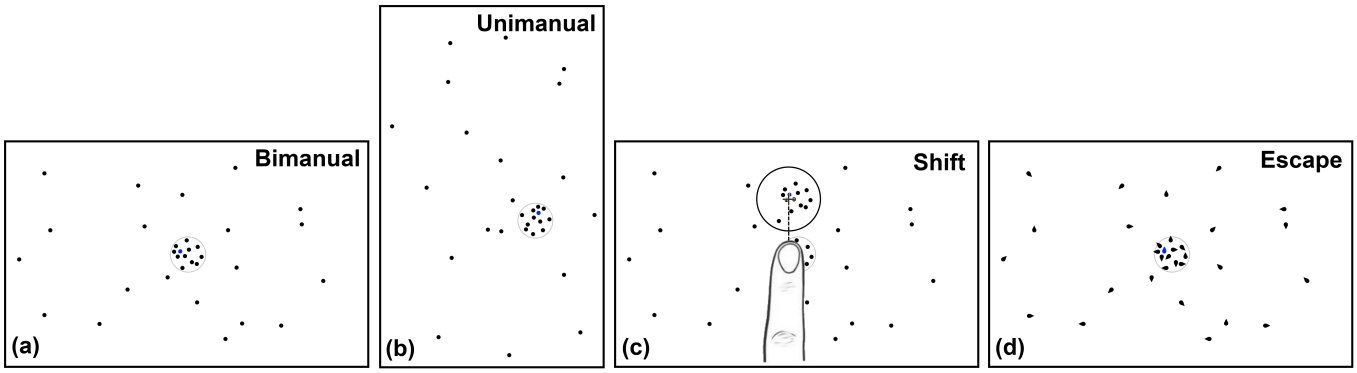

Figure 5. Cluster placements for different operation modes: (a) Bimanual mode and (b) Unimanual mode; and illustrations of the display style of two alternative selection techniques: (c) Shift [23] and (d) Escape [25].

The main reason for selecting *Shift*, *Escape* and *Bubble* as the methods for comparison is that all three methods, like *LinearDragger*, involve only single touch-drag-release operation. Note that when using *Bubble* on a touch input device, the user may drag the touching finger to highlight different targets (i.e., the target closest to the touch point), even though the highlighted target may be occluded by the touching finger. On the other hand, there are techniques that involve multiple operations, such as *TapTap* [21], which involves separated zooming and selection operation and is shown to outperform *Shift*. Although it would be interesting to compare *LinearDragger* with this type of methods, the main focus of this user study is to evaluate the performance of the target selection methods using the same touch-drag-release operation, and we leave this issue as a future work.

To better evaluate the performance in different operating scenarios, we also introduced the operation mode as one of the factors of the experiment. Participants were instructed to use both hands (bimanual mode) or single hand (unimanual mode) to perform the selection tasks. Specifically, in bimanual mode, participants need to hold the device with the non-dominant hand and perform selection with the dominant hand. In unimanual mode, participants need to hold the device and perform the tasks with the same hand. In both operation modes, participants were free to use any finger of the specified hand to complete the selection tasks.

Apparatus

Same as those in preliminary studies 1 and 2.

Participants

Twelve adult participants (8 females and 4 males) of age 24 to 29 were recruited. All participants happened to be right-handed and had experience in using computers, tablets and smart phones with touch screens. They were all recruited from a university (recruitment posters were posted for open recruitment). Each of them was given a gift voucher of USD13 for participating in the user study.

Procedure and Design

We followed the same general procedure as in preliminary study 1. Participants were required to select the desired target highlighted in blue from a cluster of candidate targets. We used parameter **Count** to control the total number of the clustered targets. The clustered targets were located within a circular region of 100 pixels ($13.5mm$) in diameter, such that no targets overlapped each other. The circular region was placed at the center of the touchscreen in the bimanual mode (Figure 5(a)) and at the bottom-right quarter ($35mm$ and $70mm$ from the right and bottom edges, respectively) in the unimanual mode (Figure 5(b)), in order to facilitate selection with the thumb of the holding hand. In addition, 20 distracter targets having the same size as the clustered targets were placed in the empty space outside the circular region.

Our experiment was a $5 \times 4 \times 3 \times 2$ within-participant design with the following factors: (1) Five techniques **TECH** for comparison: *LinearDragger*, *DirectTouch*, *Bubble*, *Escape* and *Shift*. (2) Four target sizes **Size**: 6, 12, 18 and 24 pixels ($0.8/1.6/2.4/3.2mm$). (3) Three numbers of clustered targets **Count**: 8, 12 and 16. (We set the maximum number of clustered targets to 16 because if we put too many targets into the clustered region of a diameter $13.6mm$, they would likely overlap and occlude each other.) (4) Two operation modes **Mode**: bimanual and unimanual.

Each participant performed the experiment in 2 sessions of different operation modes, each session was divided into 5 groups of different techniques. Each **TECH** group was further divided into 12 subgroups of different combinations of **Size** and **Count**. Within each subgroup, each participant needed to complete 20 selection tasks. In the experiment, **Mode** was counterbalanced, **TECH** (except *DirectTouch*) was ordered in a balanced Latin-square with *DirectTouch* always being used at the end, and the order of subgroups was randomized. A total of 28,800 selection tasks were included in the experiment. Similar to the preliminary studies, before using each operation mode and technique, participants were instructed and given a 5–10 minute warm-up session to get familiar with the technique and the tasks.

We implemented *Shift* [23] with the zero escalation time. This is because in each trial, all clustered targets were of the same small size (with a maximum size of $3.2mm$) packed within a small area. Hence, we assumed that the pop-up of the occluded area was always needed by the participants. However, no refinement or correction of the touching position was used

as we would like to provide a consistent touching point precision and control among all techniques including *DirectTouch* and *Bubble*, which are very sensitive to the touching positions. *Escape* [25] was implemented with 8-directional beak-shape targets. Note that unlike the original design in [25], we did not apply any color scheme as visual cues, as we would like to have a similar display complexity and style for all tested techniques. (All targets were in black with the desired target in blue.) Figure 5(c) shows the pop-up local copy of the occluded area and Figure 5(d) shows the same set of clustered targets with the beak-shape indicators. For *LinearDragger*, we set the size of the ROI to 52 pixels ($7.2mm$) and used the Orthogonal Dragging cancellation mechanism. In order to have a fair evaluation on *LinearDragger*, we disabled the target expansion feature (i.e., the Bubble cursor approach) in the main experiment.

Results

Selection Time

Repeated measures analysis shows a significant main effect for **TECH** ($F_{(4,44)} = 32.4, p < 0.0001; F_{(4,44)} = 48.72, p < 0.0001$), **Size** ($F_{(3,33)} = 5.0, p = 0.0022; F_{(3,33)} = 7.91, p < 0.0001$) and **Count** ($F_{(2,22)} = 3.31, p = 0.0481; F_{(2,22)} = 3.35, p = 0.0371$) on both bimanual and unimanual modes, respectively. The mean selection time was $2557ms$ for *LinearDragger*, $8336ms$ for *DirectTouch*, $3989ms$ for *Bubble*, $4384ms$ for *Escape* and $3574ms$ for *Shift*. The following significant interactions are also observed: **TECH** $\times$ **Size** ($F_{(12,132)} = 2.72, p = 0.0021; F_{(12,132)} = 2.68, p < 0.0001$) and **TECH** $\times$ **Count** ($F_{(8,88)} = 2.54, p = 0.0113; F_{(8,88)} = 2.30, p = 0.0230$) for bimanual mode and unimanual mode, respectively.

Figures 6 and 7 show the mean selection times of different techniques grouped by **Size** and **Count**, using bimanual mode and unimanual mode, respectively. We can observe that *LinearDragger* had the shortest mean selection time in most of the cases. *DirectTouch* had the worst performance among all selection techniques, particularly when the target size was small. We found that it was almost impossible for participants to select targets in size of 6 pixels (0.8 mm) using *DirectTouch*. Hence, we added a maximum movement time for each selection task $30s$ in our setting) to ensure that participants could complete the sessions within a reasonable amount of time, and a task would be terminated if it could not be finish in time. Since more that 80% of the selection tasks using *DirectTouch* and with target size of 6 pixels ($0.8mm$) could not be finished successfully in time, we decided to skip this combination of *DirectTouch* in the computation of average movement time and error rate. This does not affect the overall evaluation as *DirectTouch* only served as the baseline in our experiment. To gain a thorough understanding of the measurement, the Tukey HSD post-hoc tests are conducted to evaluate the significance in the performance gaps. Post-hoc tests analysis the significance in the performance difference among different techniques. We present the post-hoc test result in the following manner. A technique, T_1, has a rank higher than that of another technique, T_2, if and only if the post-hoc test shows that T_1 significantly outperforms T_2 under the given scenario.

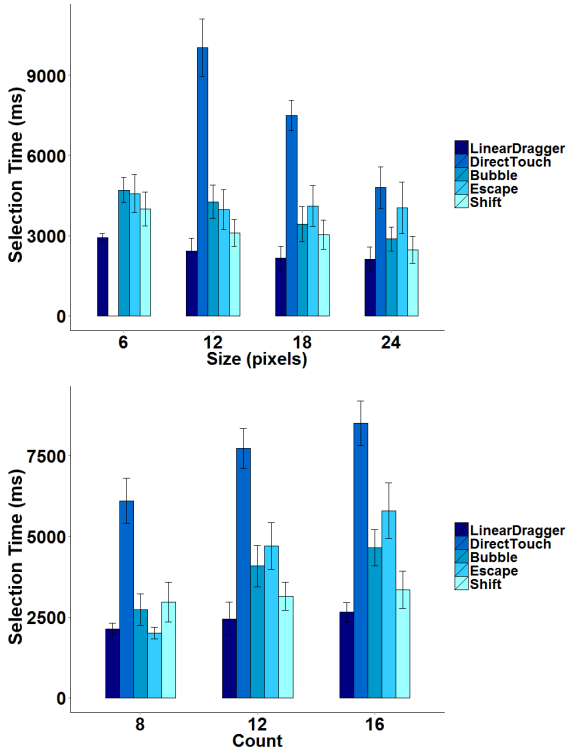

Figure 6. The selection times of different techniques grouped by Size (up) and Count (down), for the bimanual operation mode. The error bars are at the 95% confidential level.

The post-hoc test result shown in Table 1 reveals that *LinearDragger* was significantly faster than all other methods, for target sizes smaller than 18 pixels ($2.4mm$). When the target size was set to 24 pixels ($3.2mm$), there was no significant difference in the mean selection time among *LinearDragger*, *Bubble* and *Shift* in the bimanual operation mode. The post-hoc test result shown in Table 2 reveals that both *LinearDragger* and *Escape* performed the best and were significantly faster than *DirectTouch*, *Bubble* and *Shift* when **Count** was set to 8, in both operation modes. However, the performance of *Escape* degraded dramatically when **Count** increased.

Error Rate

Repeated measures analysis of variance shows that the error rate was significantly affected by **TECH** ($F_{(4,44)} = 8.31, p < 0.0001; F_{(4,44)} = 10.82, p < 0.0001$) for bimanual mode and unimanual mode, respectively. The mean error rate was 7.2% for *LinearDragger*, 35.3% for *DirectTouch*, 8.5% for *Bubble*, 29.1% for *Escape* and 9.7% for *Shift*. The significant interactions **TECH**$\times$ **Count**($F_{(8,88)} = 5.42, p < 0.0001; F_{(8,88)} = 4.93, p < 0.0001$) are also observed for bimanual and unimanual modes, respectively.

Discussion

By comparing Figures 6 and 7, we observe that target selection in unimanual mode took much more time than in bimanual mode. This can be easily explained as target selection us-

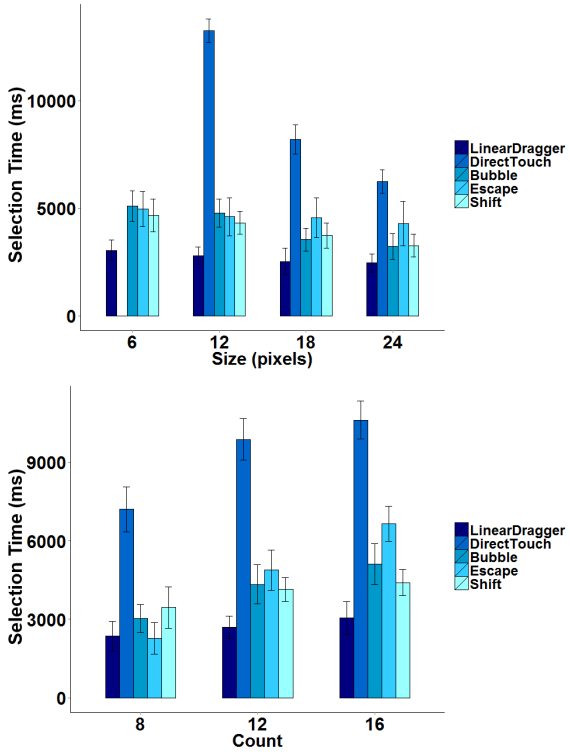

Figure 7. The selection times of different techniques grouped by Size (up) and Count (down), for the unimanual operation mode. The error bars are at the 95% confidential level.

ing both hands provides stabler control over the device than using a single hand. Besides, this effect is especially significant for *Shift* since it requires precise touch control while other techniques (except *DirectTouch*) can tolerate a noisier touch input.

From the post-hoc test result shown in Table 1, we observe that the performance of *Shift* is correlated with the target size – when the target size increased, the selection time of *Shift* decreased significantly. This is due to the fact that the operation of *Shift* is similar to *DirectTouch* but without the occlusion problem. Thus, its performance is directly related to the effective size of the target.

From the post-hoc test result shown in Table 2, we observe that *Escape* relies on assigning distinguishable swiping directions to the crowded targets. When **Count** becomes greater than the number of possible swiping directions, multiple targets in the cluster will be assigned with the same swiping direction, making them more difficult to be selected. On the other hand, as both *DirectTouch* and *Bubble* do not resolve the occlusion problem, they incurred a considerable amount of selection errors. As excepted, the performance of *Bubble* degraded with dense target distributions, since the targets' effective areas become smaller in denser cluster and are more often occluded by the finger tip.

Qualitative Results

Participants were asked to rank the techniques by subjective preference in a post-hoc questionnaire. Eight participants ranked *LinearDragger* as their preferred technique, while four ranked *Shift* as their preferred technique and *LinearDragger* as second. Most participants commented that they liked *Escape* for Count less than 8 but not very helpful for larger cluster sizes.

Tukey HSD test ($\alpha = 0.05$)						
Mode	Size	Rank				
		LG	DT	B	E	S
Bimanual	6	1	-	3	3	2
	12	1	4	3	3	2
	18	1	4	2	3	2
	24	1	3	1	2	1
Unimanual	6	1	-	2	2	2
	12	1	3	2	2	2
	18	1	4	2	3	2
	24	1	4	2	3	2

LG = *LinearDragger*, DT = *DirectTouch*, B = *Bubble*, E = *Escape*, S = *Shift*

Table 1. Significant differences on mean selection time among TECH by Size, for bimanual (top) and unimanual (bottom) modes.

Tukey HSD test ($\alpha = 0.05$)						
Mode	Count	Rank				
		LG	DT	B	E	S
Bimanual	8	1	3	2	1	2
	12	1	4	3	3	2
	16	1	4	3	3	2
Unimanual	8	1	3	2	1	2
	12	1	4	2	3	2
	16	1	5	3	4	2

LG = *LinearDragger*, DT = *DirectTouch*, B = *Bubble*, E = *Escape*, S = *Shift*

Table 2. Significant differences on mean selection time among TECH by Count, for bimanual (top) and unimanual (bottom) modes.

CONCLUSION

This paper introduces *LinearDragger*, a new one-finger target acquisition technique for small and clustered targets. Our key contribution is a mapping approach to convert the 2D targeting selection problem to a 1D selection problem, with a simple touch-drag-release operation. Our technique gives constant effective width for all candidate targets, which is independent of the target distribution. This leads to a predictable and controllable selection interface. Results of our controlled experiments show that *LinearDragger* is a promising target acquisition technique for touch devices, and performs better than existing popular selection techniques that involve a single-touch operation, in terms of both selection time and error rate.

Limitations

LinearDragger may suffer from three limitations. First, as it is designed for acquiring a target from a predefined set of discrete targets, it may not be able to support selection of arbitrary screen points in a continuous 2D space. This may be an interesting issue for further research. Second, as *LinearDragger* overwrites the original dragging operation, it may not naturally support object dragging in some applications. However this can be easily addressed by adopting a tap-and-a-half operation to enter the object dragging mode after an object is being selected. Third, it is difficult to apply *LinearDragger* at the OS level without modifying individual client applications. However, as most recent OSs support rich touch APIs

and dynamic reflection of UI elements from client applications, we believe that the proposed method can be integrated to the applications by introducing an additional global touch event hooking at the OS level, to convert appropriate dragging events into selection operations.

Future Work

As a future work, we would like to compare *LinearDragger* with other techniques involving different genres of touch operations, such as *TapTap* [21], which involves multiple tapping operations and is shown to outperform *Shift*, and to study user preferences about touch operations and other aspects. We would like to apply and evaluate *LinearDragger* in other application scenarios, such as multiple object selections. We would also like to investigate the possibility of applying *LinearDragger* in context-aware applications, such as an integrated interface including context menus.

ACKNOWLEDGEMENTS

We thank all the reviewers of this paper for the insightful comments and constructive suggestions. The work described in this paper was partially supported by two grants from City University of Hong Kong (CityU Reference No.: 7003056 and 7004155), and two GRF grants from the RGC of Hong Kong (RGC Reference No.: CityU 116010 and CityU 115112).

REFERENCES

1. Albinsson, P.-A., and Zhai, S. High precision touch screen interaction. In *Proc. ACM SIGCHI* (2003), 105–112.
2. Baudisch, P., Zotov, A., Cutrell, E., and Hinckley, K. Starburst: a target expansion algorithm for non-uniform target distributions. In *Proc. Working Conference on Advanced Visual Interfaces* (2008), 129–137.
3. Benko, H., Wilson, A., and Baudisch, P. Precise selection techniques for multi-touch screens. In *Proc. ACM SIGCHI* (2006), 1263–1272.
4. Bi, X., Li, Y., and Zhai, S. Fitts law: Modeling finger touch with fitts' law. In *Proc. ACM SIGCHI* (2013), 1363–1372.
5. Blanch, R., Guiard, Y., and Beaudouin-Lafon, M. Semantic pointing: improving target acquisition with control-display ratio adaptation. In *Proc. ACM SIGCHI* (2004), 519–526.
6. Chapuis, O., Labrune, J.-B., and Pietriga, E. Dynaspot: speed-dependent area cursor. In *Proc. ACM SIGCHI* (2009), 1391–1400.
7. Fekete, J.-D., and Plaisant, C. Excentric labeling: Dynamic neighborhood labeling for data visualization. In *Proc. ACM SIGCHI* (1999), 512–519.
8. Findlater, L., Jansen, A., Shinohara, K., Dixon, M., Kamb, P., Rakita, J., and Wobbrock, J. Enhanced area cursors: reducing fine pointing demands for people with motor impairments. In *Proc. ACM UIST* (2010), 153–162.
9. Grossman, T., and Balakrishnan, R. The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. In *Proc. ACM SIGCHI* (2005), 281–290.
10. Gutwin, C. Improving focus targeting in interactive fisheye views. In *Proc. ACM SIGCHI* (2002), 267–274.
11. Karlson, A., and Bederson, B. Thumbspace: generalized one-handed input for touchscreen-based mobile devices. In *Proc. INTERACT*, Springer (2007), 324–338.
12. Käser, D., Agrawala, M., and Pauly, M. Fingerglass: efficient multiscale interaction on multitouch screens. In *Proc. ACM SIGCHI* (2011), 1601–1610.
13. Kopper, R., Bacim, F., and Bowman, D. Rapid and accurate 3D selection by progressive refinement. In *Proc.. IEEE Symp. 3D User Interfaces* (2011), 67–74.
14. Moscovich, T. Contact area interaction with sliding widgets. In *Proc. ACM UIST* (2009), 13–22.
15. Moscovich, T., and Hughes, J. Multi-finger cursor techniques. In *Proc. Graphics Interface* (2006), 1–7.
16. Olwal, A., Feiner, S., and Heyman, S. Rubbing and tapping for precise and rapid selection on touch-screen displays. In *Proc. ACM SIGCHI* (2008), 295–304.
17. Parhi, P., Karlson, A., and Bederson, B. Target size study for one-handed thumb use on small touchscreen devices. In *Proc. MobileHCI* (2006), 203–210.
18. Parker, J., Mandryk, R., Nunes, M., and Inkpen, K. Improving target acquisition for pointing input on tabletop displays. In *Proc. INTERACT* (2005), 80–93.
19. Pietriga, E., and Appert, C. Sigma lenses: focus-context transitions combining space, time and translucence. In *Proc. ACM SIGCHI* (2008), 1343–1352.
20. Potter, R., Weldon, L., and Shneiderman, B. Improving the accuracy of touch screens: an experimental evaluation of three strategies. In *Proc. ACM SIGCHI* (1988), 27–32.
21. Roudaut, A., Huot, S., and Lecolinet, E. Taptap and magstick: improving one-handed target acquisition on small touch-screens. In *Proc. Working Conference on Advanced Visual Interfaces* (2008), 146–153.
22. Sears, A., and Shneiderman, B. High precision touchscreens: design strategies and comparisons with a mouse. *International Journal of Man-Machince Studies 34*, 4 (Apr. 1991), 593–613.
23. Vogel, D., and Baudisch, P. Shift: a technique for operating pen-based interfaces using touch. In *Proc. ACM SIGCHI* (2007), 657–666.
24. Worden, A., Walker, N., Bharat, K., and Hudson, S. Making computers easier for older adults to use: area cursors and sticky icons. In *Proc. ACM SIGCHI* (1997), 266–271.
25. Yatani, K., Partridge, K., Bern, M., and Newman, M. Escape: a target selection technique using visually-cued gestures. In *Proc. ACM SIGCHI* (2008), 285–294.