2D-Dragger: Unified Touch-based Target Acquisition with Constant Effective Width

Qingkun Su[1] **Oscar Kin-Chung Au**[2] **Pengfei Xu**[3,1*] **Hongbo Fu**[2] **Chiew-Lan Tai**[1]

[1]Department of Computer Science and Engineering, HKUST
[2]School of Creative Media, City University of Hong Kong
[3]College of Computer Science and Software Engineering, Shenzhen University

ABSTRACT

In this work we introduce *2D-Dragger*, a unified touch-based target acquisition technique that enables easy access to small targets in dense regions or distant targets on screens of various sizes. The effective width of a target is constant with our tool, allowing a fixed scale of finger movement for capturing a new target. Our tool is thus insensitive to the distribution and size of the selectable targets, and consistently works well for screens of different sizes, from mobile to wall-sized screens. Our user studies show that overall *2D-Dragger* performs the best compared to the state-of-the-art techniques for selecting both near and distant targets of various sizes in different densities.

Author Keywords

Target acquisition; accessibility; effective width; touch input

ACM Classification Keywords

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

INTRODUCTION

Touch-based interaction is very convenient and intuitive to perform, but suffers from various accessibility problems. First, small targets especially in a dense region, are difficult to access accurately due to the well-known fat finger problem, which causes both inaccurate input and undesired occlusion. This problem is more obvious on smaller touch screens, e.g., of wearable devices. On the other hand, on big touch screens (e.g., table-size, wall-size), distant objects are not accessible by direct touch. A similar problem also exists with mobile devices when using single-hand interaction. The thumb becomes the only finger for interaction which is rather short in the context of bigger mobile touch screens.

The *small-target* and *distant-target* acquisition problems have been studied extensively. However, most of the existing techniques focus only on one of the two problems. For example, *Shift* [24] and *LinearDragger* [2] are two techniques for selecting small targets on mobile touch screens. *ThumbSpace* [14] and *BezelCursor* [15, 16] are designed to facilitate selection of distant objects in the scenario of single-handed mobile interaction. Many techniques such as *Vacuum* [5] and *Drag-and-Pick* [3] have been proposed to access distant objects on large displays. *MagStick* [22] is one of the very few methods that aim to resolve both the small- and distance-target problems, but it is designed for small screens (e.g., mobile phones) and does not scale well to larger displays (e.g., tablet/tabletop screens).

Our work is inspired by *LinearDragger* [2], a recent technique that maps the targets in a local region under the touching finger into a 1D list and allows users to select one of the targets using a linear dragging operation. Since each target in such a local region is associated with a fixed effective width, it is capable of selecting a small target in a dense cluster. However, this method cannot access targets outside the local region, let alone distant objects, since representing numerous 2D elements in a single 1D list is not effective.

Our technique, which we call *2D-Dragger*, is a unified touch-based target acquisition technique that enables easy access to both small and distant targets on screens of various sizes, thus providing users a consistent experience in different devices and application scenarios. As illustrated in Figure 1, we skip empty spaces between selectable targets and use a pointing-like mechanism to select nearby targets, instead of packing all elements into a single 1D list. This enables easy access to distant objects even on large-size screens. *2D-Dragger* uses a fixed scale of finger movement to capture a new target so that each target has a constant effective width, thus making it work equally well for different target sizes and distribution densities. *2D-Dragger* does not cause any finger occlusion as the user can start the selection at any location on the screen. Since no virtual cursor is needed, our tool also introduces no auxiliary visual elements.

We have conducted a controlled user study to evaluate the performance of *2D-Dragger* under different conditions, and found that our technique performed better than *LinearDragger*, *ObjectPointing*, and *Bubble+Touchpad* in terms of selection time and/or error rate for reachable selection (i.e., selecting targets within the reach of a finger or arm). *2D-Dragger* also outperformed *ObjectPointing* and *MagStick* in terms of efficiency or accuracy under the remote selection scenario (i.e., selecting targets beyond the reach of a finger or arm).

RELATED WORK

*Corresponding author: Pengfei Xu (pengfeixu@szu.edu.cn)

MobileHCI '16, September 06-09, 2016, Florence, Italy.

http://dx.doi.org/10.1145/2935334.2935339

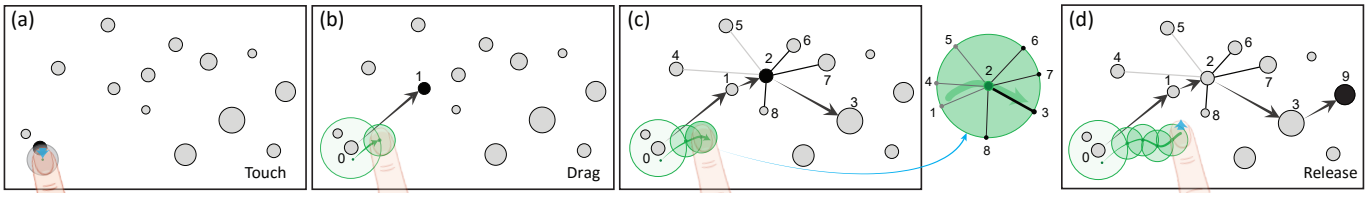

Figure 1. Overview of *2D-Dragger*. Our technique uses a simple touch-and-drag operation to achieve efficient target acquisition. (a) When the user places a finger on the screen, a selectable target (if any) near the contact point within a small region of interest (gray circle) becomes the first captured target (solid black element 0). (b) Meanwhile a circular effective area (the left lighter green circle; slightly bigger for the firstly captured element) is defined at the captured element 0. The user can then capture a nearby target (element 1) by dragging the contact finger out from the effective area in the direction towards the new target. The right green circle is the effective area of element 1. (c) When a new target (element 2) is captured, a new circular effective area (the rightmost green circle) is placed at the current finger location. The user can then continuously capture the next nearby target by keep dragging the contact finger in the user's intended direction. (d) Once the desired target is captured, the user can lift the contact finger to confirm the selection. The rightmost green circle represents the effective area of the current selection. Note that the green circles here are invisible to users.

We focus our discussions here mainly on the target acquisition techniques specifically designed for touch-based interfaces. Many techniques have been proposed to solve the inaccuracy and occlusion issues caused by the fat-finger problem. For example, *Offset Cursor* [21] and *Shift* [24] focus on solving the occlusion problem. The accuracy and/or the overall performance of acquisition can be improved, for example, by enlarging the effective size of small targets (e.g., [20, 4]) or their visual size via zooming (e.g., [22, 19]), or using multiple taps on small targets [1]. However, all the above techniques become ineffective or even infeasible for selection of distant objects, which are beyond the reach distance (e.g., of a thumb for single-handed mobile interaction, or of an arm for large screens). For example, both *Shift* and *TapTap* [22] require the initial touching point to be in close proximity to an element of interest.

Several techniques have been proposed to address the remote selection problem caused by single-handed interaction of mobile devices with screen sizes bigger than the reach limit of the interaction thumb. This is generally achieved by mapping the thumb motion to cursor motion in various ways. The *MagStick* technique [22] uses the first contact point as a reference point and the user can then drag the thumb to control a virtual cursor such that the virtual cursor is moved in the opposite direction with respect to the reference point. Due to the 1:1 correspondence between motor and display movement, the reach limit is at most doubled by *MagStick*. In other words, *MagStick* only partially solves the distant-target acquisition problem. The *ThumbSpace* technique [14] allows the user to define a small input area within the thumb's reach and maps this input area to the entire screen space so that the user can control a virtual cursor by using this input area as a touch pad. While *ThumbSpace* works well for large screens, it lacks precise touch control to acquire small targets. In addition, *ThumbSpace* still suffers from the occlusion problem.

The problem of how to access distant objects on areas beyond an arm's reach of the user has also been explicitly studied. For example, *Vacuum* [5] uses a circular widget with a user controllable arc of influence and brings far away objects inside this influence arc closer to the widget's center in the form of proxies for selection and manipulation. Forlines et al. present *HybridPointing* [11], which relies on a circular trailing widget and allows fluid switching between absolute and relative pointing with a direct input device. In contrast, our technique does not involve any explicit mode switching. *Gesture Select* [8] is designed for selecting remote targets on large screens through user-drawn initial and continuation marks, instead of pointing-based selection. While these techniques might be adapted for single-handed mobile interaction, how to make them effective for selecting small targets in a dense region is not an easy problem.

One possible solution to both the small- and distant-target acquisition problems is to adopt cursor-based techniques (e.g., [26, 13, 7, 9, 12, 10]) by replacing mouse input with a virtual touchpad input. For example, *Bubble Cursor* [12], which dynamically resizes the activation area of the cursor depending on the proximity of surrounding targets, might be adapted. With *Bubble Cursor*, the effective width of targets is dependent on target distribution and thus is small when the distribution is dense. Our technique bears close resemblance to *ObjectPointing* [13], in the sense that both techniques aim to skip empty spaces between selectable targets and at any time only one target is captured. Their main difference is that, in *ObjectPointing*, the effective widths are the sizes of the selectable targets so that the cursor moves only within the targets and is visible to the user. Such a design would lead to unpredictable control, since the switching of the captured target would be too fast (slow) if the target sizes are too small (big). A more detailed comparison between *2D-Dragger* and touch-based *Bubble Cursor* and *ObjectPointing* is conducted in our user study.

There are many methods using dynamic effective width (e.g., *Bubble Cursor* [12], *DynaSpot* [9], *ObjectPointing* [13]), which usually assign small effective width to dense targets. Therefore, they require very accurate input to select desired targets in a dense region. Another method that uses dynamic effective width is *Bayesian Touch* [6], which conceptualizes touch inputs as an uncertain process and improves the accuracy using Bayesian touch criterion. However, this approach is designed for direct-touch-based selection and how to extend it for remote target selection is unclear.

DESIGN RATIONALE

The design of *2D-Dragger* is driven by the following set of guidelines:

A unified interface. A normal user nowadays often has access to multiple touch devices (e.g., smartphone and touch-enabled TV display), which are equipped with displays of different sizes and resolutions. Providing a single unified target selection technique can certainly enhance user experience when using such touch devices with different specifications.

Skipping empty space. For sparsely distributed elements, there are large empty gaps between elements. Skipping such empty spaces will greatly aid the task of element selection on large-size screens. This also generally shortens the selection time according to Fitts' law reference.

Constant effective width. A constant effective width is desirable for accessing elements of various sizes. It avoids error-prone selection for small-size elements, and reduces moving distance for large-sized elements.

Supporting both direct and indirect selection. Direct tap no doubt is the fastest way to access an element that is within reach. However, for effective selection of small-size elements (due to undesired occlusions) or remote elements (out of users' reach), indirect selection might be preferred.

USER INTERFACE

As illustrated in Figure 1, *2D-Dragger* uses a simple touch-and-drag operation to achieve efficient target acquisition. When the user places a finger (e.g., a thumb for single-handed interaction of mobile devices) on the screen, if there is any selectable target near the contact point, this target will become the currently captured target (Figure 1a). Otherwise (i.e., the finger touches an empty space), the contact point is used as a virtual starting point. Meanwhile a small circular effective area is defined at the initial contact point. The user can then capture a nearby target by dragging the contact finger out from the effective area in the corresponding direction, i.e., the approximated direction from the currently captured target to the next target (Figure 1b). When a new target is captured, a new circular effective area is placed at the current finger location. The user can continue to capture the next nearby target by keep dragging the contact finger in the user's intended direction (Figure 1c). Once the desired target is captured, the user can lift the contact finger to confirm the selection (Figure 1d). Since *2D-Dragger* decouples the motion space and visual space, it avoids finger occlusion naturally. The user can place the finger near the target rather than on it, and then select it by our regular dragging control.

A key feature of *2D-Dragger* is that, while dragging to access a target, it associates every selectable item with a circular effective area of constant width (Figure 1c). Note that the effective area is always defined and centered at the current finger location when a new target is captured. This ensures that only a fixed scale of finger movement (to move outside the circular effective area) is required to capture a new target. This feature enables a regular dragging control for users to capture any nearby target from the currently captured item. Therefore, the control of our method is insensitive to the distance between and size of the selectable targets. We adopt a small effective width so that the user can easily access distant targets with only a short dragging motion.

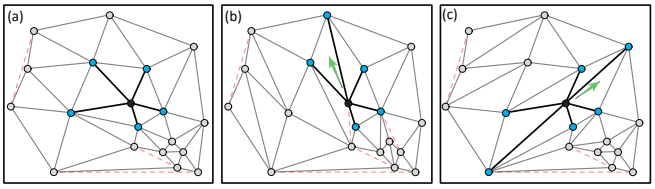

Figure 2. Illustration of dynamic Delaunay triangulation (DT). (a) Static DT; (b) and (c) In our dynamic DT, the elements in the finger's moving direction (arrows in green) are more likely to be the selection candidates to better reflect the user's intention.

Our design leads to a less error-prone selection experience. Since the effective area is always centered at the current touch point when a new target is captured, the user can safely select the currently captured target by releasing the touching finger. The centered effective area ensures that the selection operation would not be easily affected by the noisy input caused by the finger releasing action. In contrast, *LinearDragger* defines the effective area with a simple linear partition. Therefore, it is more likely to introduce selection errors when the touch point is close to the boundary of the effective area of the current target, any small movement of the touch finger could cause an unwanted switching to the target before or after the current target.

Our method is more general and adaptable to different target distributions than *LinearDragger* due to one major difference. *LinearDragger* only allows a fixed browsing direction (defined by the initial dragging direction) and therefore is not directly applicable to targets in a large region, since it would cause unpredictable target switching in the direction orthogonal to the dragging direction. *LinearDragger* requires $O(n)$ time on average to reach a desired target because of its linear search property. In contrast, *2D-Dragger* allows users to change the browsing direction during interaction and only the targets in the moving direction will be captured. Therefore, in theory, only $O(\sqrt{n})$ time on average is needed for *2D-Dragger* to capture a desired target.

We integrate our dragging interface to the classical direct touch method by associating a bigger effective area to the first captured target (element 0 in Figure 1b) if the user directly places the finger over it (i.e., not in an empty space). The user can then slightly drag the finger away from the first contact point to expose the captured target before confirming the selection, or the user could use our dragging control to select other nearby targets. This design can reduce the error rate of direct touch by allowing the user to correct the captured target if the first captured target is not the desired one.

IMPLEMENTATION

ROI and First Captured Target

When the user places a finger tip on the touch screen, a local region of interest (ROI, gray circle in Figure 1a) is defined at the first touch point. If there is any selectable target within this ROI, the closest one is picked as the initial captured target. If no such target is found, we temporarily add a virtual

starting point at the first touch point. In both cases the user can then drag the touching finger to capture a nearby real target, or accept the currently captured target (if any) by releasing the touching finger. Note that the virtual starting point will be removed after the user has captured the first nearby target. We empirically set the radius of the ROI to 3.6mm (i.e., 26 pixels for our testing device), which is small enough while still selectable by human finger.

Determining Nearby Targets

Recall that the user can capture any nearby target from the currently captured one by dragging the touching finger in the corresponding direction. To define the set of nearby targets, we compute a Delaunay triangulation (DT), which takes the centers of the selectable targets as the input vertices. As illustrated in Figure 2, the candidate set of the nearby targets (dots in blue) of a given target (dot in black) is their *one-ring neighborhood* in the computed DT.

We assume that the finger's moving direction indicates the user's intention, and the elements in this direction are thus more likely to be the selection candidates. To facilitate better orientation control of our interface and allow easier capture of the targets in the finger's moving direction, instead of a static DT, we compute a dynamic DT, which is driven by the moving direction of the finger. Specifically, when computing the dynamic DT, we define the distance between two targets p and q as follows:

$$\text{dist}(p, q) = \|\lambda \, \text{proj}_{\mathbf{v}}(\mathbf{d}_{pq}) + \text{proj}_{\mathbf{v}_\perp}(\mathbf{d}_{pq})\|, \qquad (1)$$

where $\mathbf{d}_{pq}$ is the displacement vector from p to q, and $\mathbf{v}$ is the normalized velocity vector of the finger's movement, obtained using the same method in [23]. $\mathbf{v}_\perp$ is the unit vector perpendicular to $\mathbf{v}$ and $\text{proj}_{\mathbf{v}}(\cdot)$ is the projection function that projects the input vector to $\mathbf{v}$. λ is a parameter in the range $[0, 1]$ to decide how much the triangulation space should be scaled along the moving direction. $\lambda = 1$ for the traditional Delaunay triangulation, i.e., $\text{dist}(p, q) = \|\mathbf{d}_{pq}\|$. With this distance function, when the displacement vector $\mathbf{d}_{pq}$ is perpendicular to the moving direction $\mathbf{v}$, $\text{dist}(p, q) = \|\mathbf{d}_{pq}\|$, without any change. If $\mathbf{d}_{pq}$ is parallel with $\mathbf{v}$, $\text{dist}(p, q) = \lambda|\mathbf{d}_{pq}\|$, which becomes smaller and is scaled with the ratio λ. In our current implementation we set λ as 0.5. Two examples of the dynamic DT are shown in Figure 2 (b) and (c).

There may exist slim obtuse triangles in the computed DT. The neighboring relationship represented by the longest edges (red dotted edges in Figure 2) of these triangles, which we call the *weak edges*, is not desirable as such edges may cause undesired target switching during finger dragging. We thus simply remove them from the computed DT. We flag an edge e as a weak edge if its length is more than $0.95(a + b)$, where a and b are the lengths of the opposite edges of any triangle sharing the edge e.

Determining Next Captured Target

After identifying the set of nearby targets $Q = \{q_i\}$ of the currently captured target p, we need to find the best one $q \in Q$ according to the finger's moving direction. Our first attempt was to simply choose the nearby target with the best matching direction, i.e., $q = \arg\min_{q_i} \theta_i$, where θ_i represents the angle between $\mathbf{d}_{pq_i}$ and $\mathbf{v}$. However, we observed that this often produces undesired switching between targets, due to the user's inability of precisely specifying their desired directions. To address this difficulty, we propose an approach which is more tolerant of imprecise user input. First, we find a subset Q' of Q that roughly matches the finger's move direction. Specifically, Q' is defined as $\{q_i|\theta_i < \alpha\}$, where α is the deviation angle that represents how precise the user can specify a given direction by moving a finger. The value of α is determined by our preliminary study. There might exist two or more elements in Q', especially when the current selection is surrounded by many potential targets. Simply switching to the one with the smallest angle would not be reliable , since it is not easy to specify accurate switching direction angles by finger. Instead, the target $q \in Q'$ with the shortest distance to the currently captured target is picked as the next captured target, to avoid selecting an unintended target with a smaller angle but a larger distance. If Q' is empty, we choose the target $q \in Q$ that best matches the moving direction:

$$q = \begin{cases} \underset{q_i \in Q'}{\text{argmin}} \, \|\mathbf{d}_{pq_i}\| & Q' \neq \emptyset \\ \underset{q_i \in Q}{\text{argmin}} \, \theta_i & Q' = \emptyset. \end{cases} \qquad (2)$$

To avoid target switching in the backward direction, we simply disable switching to a target q_i if it is behind the current moving direction (i.e., when $\theta_i > 90°$).

Selection Cancelation

We have implemented and tested two cancelation methods. First, move back the finger to the first contact point to cancel the selection. Second, introduce virtual targets at a screen boundary so that the user can drag the finger to the boundary to cancel the selection. We included both methods in our system.

PRELIMINARY STUDY

The performance of *2D-Dragger* is affected by two user-dependent parameters, i.e., deviation angle α and effective width w. We designed two preliminary studies to determine the values of these two parameters.

Apparatus

The experiment was conducted on an Acer W3-810 tablet computer, with 1.80 GHz Intel Atom CPU, 2.00 GB RAM and an 8.1″ (20.6cm) touch screen running MS Windows 8. The screen is of 17.5cm × 11cm and has a resolution of 1280×800 (73.6 pixels/cm).

Participants

We recruited 8 university students (4 males, 4 females) to participate in both preliminary studies. All of them were right-handed and had extensive experience in using touch devices, such as tablets or smart phones.

Preliminary Study I: Determining Deviation Angle

Task Design. In each task of this study, the participants were given a reference direction and they were required to specify the same direction by dragging their finger on the touch

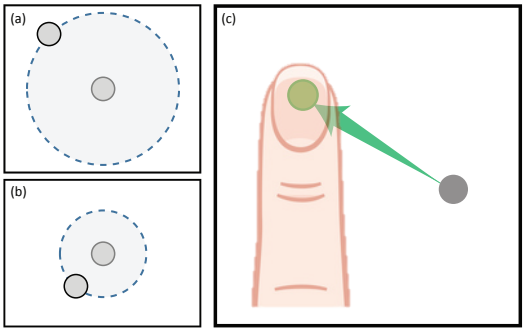

Figure 3. Illustration of preliminary study I. (a, b) In each task of this study, participants were given a reference direction, which was visualized as two points. The distance between the two points varies in different tasks. (c) The participants were asked to specify the same direction by dragging their finger on touch screen.

screen (see Figure 3). The reference direction was illustrated by two small circles (of radius 6 pixels) on the screen, with the one at the screen center as the starting point. The proximity between the two circles was considered as a controlled factor, with the settings {20px, 30px, 40px} to simulate different densities of elements in a region. In total 12 different directions were evenly sampled and tested in the study, and for each configuration 3 identical tasks were tested. The order of the tasks in all configurations were randomized during the study. In total there were 8(USER) × 3(PROXIMITY) × 12(DIRECTION) × 3(TASK) = 864 trials were recorded. Before the user study, we explained the details of the procedure and gave each participant several minutes warm-up time to get familiar with this user interface.

Results. For each trial we recorded the deviation angle, i.e., the absolute value of the angle between the reference direction and the direction specified by the participants. The mean deviation angle μ was 12.455° with the standard deviation σ = 9.995°. Repeated measures ANOVA showed that there was no significant main effect in deviation angle ($F(11, 84) = 0.691, p = 0.743$) and different proximity settings ($F(2, 21) = 0.810, p = 0.458$) settings for tested directions. This means the moving direction accuracy is insensitive to the directions and distances between the source and target. Therefore, using a predefined constant deviation angle α would be sufficient for all configurations. Accordingly, we set the deviation angle threshold α as $\mu + \sigma$ = 22.45°, which is greater than around 83% of the deviation angles recorded in this study.

Preliminary Study II: Determining Effective Width

The effective width determines how long the finger has to move to trigger selection switching. It affects selection efficiency and accuracy. If it is set too small, switching would be very sensitive to the finger's movement and thus increase the error rate and selection time. On the other hand, if set too large, it would take longer time and distance to select a target, but with lower error rate. We conducted this preliminary study to find an effective width that has both good selection efficiency and low error rate.

Task Design. In this study the participants were asked to perform selection tasks with our technique in 4 different effective width (target diameter) configurations, i.e., 5px, 20px, 35px, and 50px. In each task, the participants needed to select a given target element from a set of randomly distributed targets. As we aimed to remove the effect of the initial touch position in this study, we restricted the first captured element to be at the center of the canvas, independent of the first touch point of the contact finger. For each configuration, 40 tasks were assigned to each participant. The participants were allowed to redo the selection when an incorrect target was selected. In this study, a total of 8 (USER) × 4 (SIZE) × 40 (TASK) = 1280 trials were performed, and the completion time, finger moving distance, and number of selection for each trial were recorded.

Results. As can be shown in the results (Figure 4), 20px was the best effective width, which overall gave the best efficiency and lowest error rate. Repeated measure ANOVA showed a significant main effect in both selection time and error rate for the tested effective width settings (selection time: $F(3, 21) = 11.784, p = 4 \times 10^{-5}$; error rate: $F(3, 21) = 14.510, p < 7 \times 10^{-6}$). Note that when the effective width was too small (5 pixels), both selection time and error rate were very large. When the effective width increased from 20 pixels to 50 pixels, the error rate did not change significantly, while the selection time increased linearly. For these effective width settings, no significant main effect was observed (selection time: $F(2, 14) = 1.544, p = 0.237$; error rate: $F(2, 14) = 0, p = 1$). We chose 20px as the best effective width, since it was among the best options, while was small enough to minimize the finger's move distance.

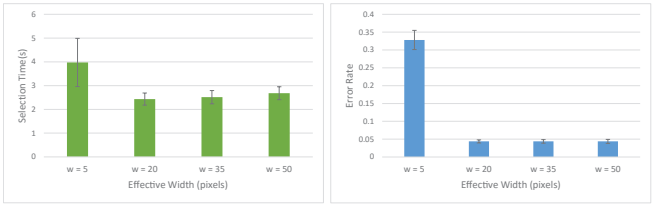

Figure 4. The selection times (left) and error rates (right) under different effective width configurations.

EVALUATION

To evaluate the effectiveness of *2D-Dragger* we compared our technique with other target selection techniques in two scenarios, namely, reachable selection (Study I) and remote selection (Study II), as illustrated in Figure 5. These two selection scenarios simulate many kinds of selection tasks on touch devices. The compared techniques were *Bubble+Touchpad*, *MagStick*, *LinearDragger*, and *ObjectPointing*. We chose the four techniques because no other technique is applicable to both reachable and remote selection. We modified some of them in our experiment for a fairer comparison. In addition, the experiments in [2] already showed that *LinearDragger* was significantly faster than Direct Touch / *Bubble* [12] / *Escape* [27] / *Shift* [24] and we compared *2D-Dragger* with *LinearDragger* in Study I (reachable selection scenario) using a similar task design. Therefore, we could indirectly compare *2D-Dragger* with these techniques using

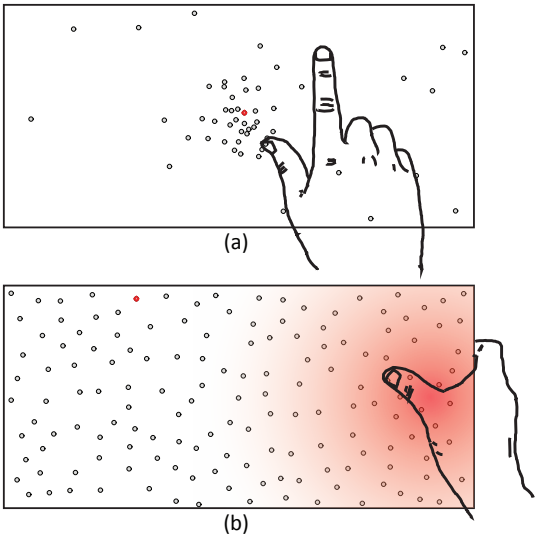

Figure 5. Two scenarios evaluated. (a) Reachable selection scenario: the user can use a finger to directly touch any position on the screen. (b) Remote selection scenario: it is difficult for the user to reach some area of the screen. The area in red roughly represents where the user's thumb can reach. The red dots are the targets to be selected.

LinearDragger as reference. A detailed description of the compared techniques is given below.

Bubble+Touchpad. *Bubble Cursor* [12] was first introduced to assist element selection with computer mouse. A simple extension of *Bubble Cursor* on touch devices was to use the contact point of the finger as the cursor [2]. This setting, however, was unable to avoid the occlusion of the finger, making the selection error-prone in dense layouts [2]. Moreover, remote selection is not possible with this setting as the nearest target is always selected. To address these problems, we combined *Bubble Cursor* with a virtual *Touchpad* [25] in our experiment. When the user touched the screen, the position of the underlying cursor was initially set the same as the contact point. When she moved her contact finger, the speed of the cursor was set to three times faster than that of the finger so that she could reach anywhere for reachable selection on the screen of our test device while still being able to do precise selection. We tested other speed parameters for our touch device, and found that (a) larger values reduced the selection time but led to harder control of the selection and thus higher error rate; (b) smaller values gave better control but increased the selection time. The speed of 3 times faster achieved a good balance between selection accuracy and efficiency.

There are many other refined touchpad-based methods [17, 18] using multiple touch actions or auxiliary devices for other selection applications, rather than hand-held touch devices. However, since our method can also be adapted to similar applications, we compared our method only to the original touch-pad techniques with special refinement for target-aware pointing.

MagStick. *MagStick* [22] is a virtual cursor technique originally designed for selection on touch devices. The user controls the cursor's movement by moving the finger in the opposite direction. Therefore *MagStick* can avoid occlusion of finger. In our experiment we followed the original implementation but with an improvement: each element is associated with a proximity area and it is selected if the cursor is within this area. We expanded the proximity area such that at any time, only one element was selected. This implementation could further reduce the effort of selecting a target element.

LinearDragger. *LinearDragger* [2] was designed for reachable selection of small elements in dense layouts. It assigns equal-width effective areas linearly to all targets within the ROI defined by the first touch point. Since it is not suitable for remote selection, we only compared it with our tool in Study I. The implementation of *LinearDragger* in our experiment was the same as that in [2].

ObjectPointing. *ObjectPointing* [13] neglects the empty regions between elements when moving the cursor. Its selection behavior is similar to *2D-Dragger*, but the effective width in *ObjectPointing* is determined by the size of the currently selected elements. In addition, its algorithm for jumping to the next element is mainly determined by the cursor's kinematics.

Apparatus and Participants

The same tablet device in the preliminary study was used here. We recruited another 10 (4 females and 6 males) university students to help with Studies I and II. They were different from those in the preliminary study. Again all of them were right-handed and had extensive experience in using touch devices.

Study I: Reachable Selection

This study aimed to simulate the reachable selection scenario, i.e., the target element is reachable by a finger. This scenario is common when operating on a small-screen smartphone, tablet held by another hand, or touch table of small size. In this study each participant was asked to hold the tablet with his/her non-dominant hand, and to perform selection operations with the dominant hand. Five techniques (TECH), including *2D-Dragger*, *Bubble+Touchpad*, *MagStick*, *LinearDragger*, and *ObjectPointing*, were tested.

Task. As illustrated in Figure 5a, each participant was required to select a specific element, highlighted in red, from a cluster of elements. All the elements including the target one were randomly distributed in a circular region of radius 60px (8.1mm) [2]. This circular region was located at the center of the canvas, whose size was 1280px × 617px. Around the cluster we randomly added extra distraction elements with a gradually reduced density. The layout of the elements was controlled by two factors: elements size and layout density. The element size (SIZE) had three settings: 3px, 6px, and 9px. We chose 3px targets because many scenarios contain such targets (dot flags on map, small-size texts in web pages or documents). The layout density (DENSITY) was represented as the number of elements in the cluster. It had three settings as well: 8, 16, and 24. For each layout configuration, 10 tasks (TASK) were tested. Totally there were 5 (TECH) × 3 (SIZE) × 3 (DENSITY) × 10 (TASK) × 10 (USER) = 4500 trials for each participant. All tasks were divided into 5

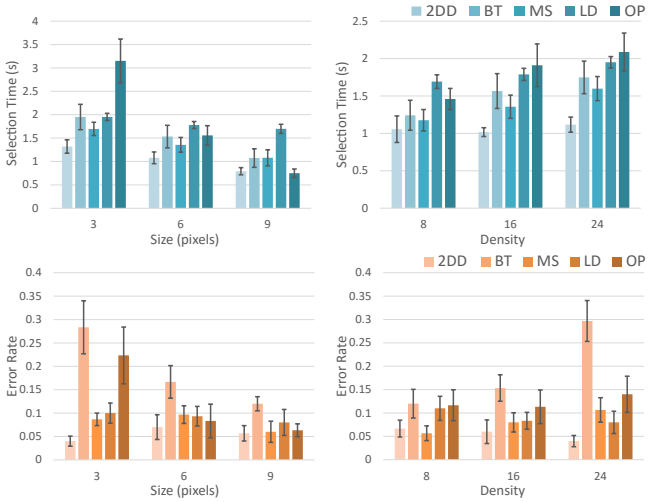

Figure 6. The selection times (top) and error rates (bottom) of different techniques(*2D-Dragger*(2DD), *Bubble+Touchpad*(BT), *MagStick*(MS), *LinearDragger*(LD), and *ObjectPointing*(OP)) grouped by Size (left) and Density (right) in User Study I. The error bars are the standard error of mean.

groups of different techniques. Each TECH group was further divided into 9 subgroups of different combinations of SIZE and DENSITY. TECH was ordered in a balanced Latin-square, while the order of subgroups was randomized.

Before the study the participants were given an instruction on how to use each technique. Then they tried each technique with similar tasks to the ones in the experiment to warm up. The instruction and warm up session lasted about 30 minutes on average. For each task, the participants were told to try their best to reduce the error while performing selection as quickly as they could. To better motivate them, we allowed each participant to have only two attempts for each task in order to avoid rapid but careless trials and errors using direct touch. After finishing a group of tasks, the participants were allowed to take a short break. On average it took each participant about 30 minutes to finish all tasks. Across all participants, the order of the techniques was counter-balanced.

Performance Measures. During the study, the following information was recorded: the selection attempts in individual trials, including successful and failed ones; the completion time for individual trials, which was the accumulation of the time of all attempts. The duration of failed attempts was also included in the completion time since they also reflected how easy to use the tested techniques were. Combined with the error rate, the completion time for individual trials give a reasonable evaluation of the overall performance of the tested techniques.

Results and Discussions. Figure 6 plots the statistics of the core information. On average it took the least time (1.063s) to finish one task with *2D-Dragger*. The average completion times per task for other techniques were: 1.519s (*Bubble+Touchpad*), 1.377s (*MagStick*), 1.810s (*LinearDragger*), and 1.819s (*ObjectPointing*). Repeated measures ANOVA showed a significant main effect in completion time for the tested techniques ($F(4, 36) = 4.453$, $p < 0.005$). The error rates of all techniques were 0.055 (*2D-Dragger*), 0.190 (*Bubble+Touchpad*), 0.081 (*MagStick*), 0.091 (*LinearDragger*), 0.123 (*ObjectPointing*). Repeated measures ANOVA showed a significant main effect in error rate for the tested techniques ($F(4, 36) = 4.999$, $p < 0.003$). We had the following findings on the interactions between different factors:

User Study I		Selection Time				Error Rate			
		BT	MS	LD	OP	BT	MS	LD	OP
SIZE	3				*	*			*
	6			*					
	9			*					
DENSITY	8			*					
	16			*	*				
	24			*	*	*			
AVG				*	*	*			

Table 1. Results of Dunnett tests for user study I. The tests were conducted as multiple comparisons, in which *2D-Dragger* was compared with all other four techniques (*Bubble+Touchpad*(BT), *MagStick*(MS), *LinearDragger*(LD), and *ObjectPointing*(OP)). An non-empty $*$ entry means there is significant difference between the specified technique and our technique for the given configuration.

- **Time**: there was a significant interaction between TECH and SIZE ($F(8, 135) = 4.937$, $p < 3 \times 10^{-5}$). Together with Figure 6, as expected, we can see that the performance of *ObjectPointing* strongly depended on the size of the elements. For other techniques, the affection of elements size was not significant. No significant interaction was found between TECH and DENSITY ($F(8, 135) = 0.497$, $p = 0.856$). This implied that all tested techniques were affected by the element density similarly.

- **Error**: the significant interaction effects were found between TECH and SIZE ($F(8, 135) = 2.470$, $p < 0.02$) as well as between TECH and DENSITY ($F(8, 135) = 2.546$, $p < 0.02$). From Figure 6, we found that the error rates of *Bubble+Touchpad* and *ObjectPointing* decreased as the element sizes increased; *Bubble+Touchpad*'s error rate increased as the element layout became denser. The error rates of other techniques were rather stable, with *2D-Dragger* the most stable.

To get a more thorough comparison, we conducted Dunnett tests as multiple comparisons, in which *2D-Dragger* was compared to all other techniques. The result was shown in Table 1. Combined with Figure 6 (bar graph for size and density configuration), we can conclude that *2D-Dragger* was never worse than any other techniques. The details are as follows:

- *Bubble+Touchpad* had achieved similar completion times to *2D-Dragger*, but its error rates were higher than those of *2D-Dragger*. Specifically, the error rates of *2D-Dragger* were significantly lower than those of *Bubble+Touchpad* when the size was small (3px) or layout was dense (=24). This is possibly because the cursor's movement is sensitive to finger's motion, especially when the target elements are very dense. So when the target elements are too small or too densely distributed, it is hard for *Bubble+Touchpad* to balance the occlusion avoidance and selection accuracy.

- Since *LinearDragger* always required dragging to a desired element from the outermost one, *2D-Dragger* was significantly faster than *LinearDragger* in most cases. This effect was even clearer when the layouts became denser.
- *2D-Dragger* was significantly faster than *ObjectPointing* when the element size was small or the layout was dense. The deficiency of *ObjectPointing* was caused by its effective width being dependent on element sizes and its element switching algorithm. When the elements were small, it became difficult for the participants to control switching with *ObjectPointing*. This was also the reason behind the significantly higher error rate of *ObjectPointing* than *2D-Dragger* in the small-size configuration. In addition, since the element switching algorithm of *ObjectPointing* does not consider the nearby relation among elements, there are many more candidate elements for switching. In dense layouts, the number of candidates becomes even larger, making it more difficult for our participants to switch to the target elements.
- In this study, *MagStick* achieved a similar performance to *2D-Dragger*, in terms of both completion time and error rate.

Study II: Remote Selection

This study aimed to simulate the scenarios where a target element is beyond the reach of a finger or arm. This situation happens when operating on a large-screen smartphone with a single hand, large-size touch table, and wall-size touch screen. The participants were required to hold the tablet with the dominant hand and use the thumb of the same hand to perform the selection operation. We tested four techniques in this study, namely, *2D-Dragger*, *Bubble+Touchpad*, *MagStick*, and *ObjectPointing*. *LinearDragger* was excluded in this study, since it requires the specification of a small set of elements for linear navigation and becomes impractical with many elements in a layout.

Task. The task was to select a desired element among distraction elements as well, but with different layout settings. For each task, the elements were randomly and uniformly distributed on a 1290px × 617px canvas, and one of them was highlighted in red as the target element (Figure 5b). Since the participants were required to hold the tablet and perform the selection with the same hand, they could not select the target by direct touch for most tasks. The layouts of the elements were controlled by two factors as well: element size and layout density. The element size had the same settings as in Study I: 3px, 6px, and 9px. The layout density was represented as the number of all elements on canvas. It had three settings: 50, 100, and 150. For each configuration, 10 tasks were tested. Totally 4 (TECH) × 3 (SIZE) × 3 (DENSITY) × 10 (TASK) × 10 (USER) = 3600 trials per participant were collected. The distribution of all tasks to the participants was similar to that in User Study I except that the size of the TECH group was 4 this time.

Since the tasks were different from the ones in Study I, the participants were given about 15 minutes to get familiar with the tasks as warm-up. The experiment was divided into four

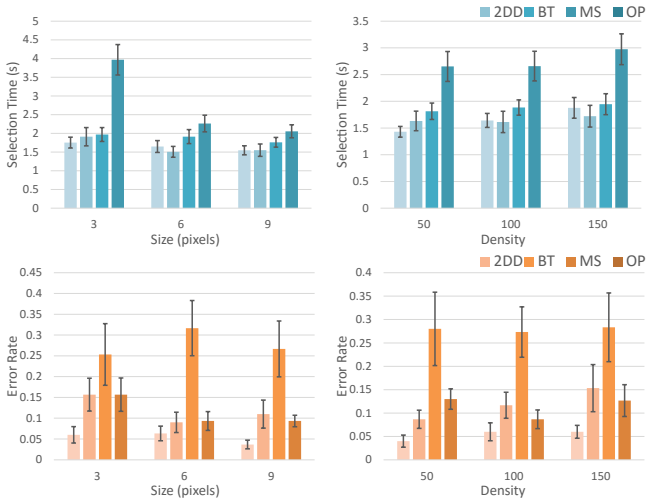

Figure 7. The selection times (top) and error rates (bottom) of different techniques (*2D-Dragger*(2DD), *Bubble+Touchpad*(BT), *MagStick*(MS), and *ObjectPointing*(OP)) grouped by Size (left) and Density (right) in User Study II. The error bars are the standard error of mean. Again space between *2D-Dragger* and *Bubble+Touchpad*

Study II		Selection Time			Error Rate		
		BT	MS	OP	BT	MS	OP
SIZE	3			*		*	
	6					*	
	9					*	
DENSITY	50			*		*	
	100			*		*	
	150			*		*	
AVG				*		*	

Table 2. Results of Dunnett tests for user study II. The tests were conducted as multiple comparisons, in which *2D-Dragger* was compared with all other three techniques (*Bubble+Touchpad*(BT), *MagStick*(MS), and *ObjectPointing*(OP)). An non-empty * entry means there is a significant difference between the specified technique and our technique in the given configuration.

groups according to the tested techniques. In each group, the order of tasks was randomized. As in Study I, the participants were also given two chances to complete each task. After finishing a group of tasks, the participants could take an optional break. Each participant completed all tasks for about 30 minutes on average.

Performance Measures. Same as those in Study I.

Results and Discussions. Figure 7 plots the statistics of the core information. On average *2D-Dragger* took the least time (1.650s) to finish each task. The average completion times per task for other techniques were: 1.656s (*Bubble+Touchpad*), 1.882s (*MagStick*), and 2.762s (*ObjectPointing*). Repeated measures ANOVA showed a significant main effect in completion time for the tested techniques ($F(3, 27) = 8.073$, $p < 0.001$). The error rates of all techniques were 0.053 (*2D-Dragger*), 0.119 (*Bubble+Touchpad*), 0.279 (*MagStick*), 0.114 (*ObjectPointing*). Repeated measures ANOVA showed a significant main effect in error rate for the tested techniques ($F(3, 27) = 6.399$, $p < 0.002$). For the interaction between different factors, we had the following findings:

- **Time**: there was a significant interaction between TECH and SIZE ($F(8, 135) = 5.183$, $p < 0.0005$). Together with Figure 7, we found that only *ObjectPointing* was strongly affected by the element size. No significant interaction was found between TECH and DENSITY ($F(8, 135) = 0.209$, $p = 0.970$). The performance of all tested techniques was not affected much by the layout density.

- **Error**: there was no significant interaction between TECH and SIZE ($F(8, 135) = 0.56$, $p = 0.759$) as well as between TECH and DENSITY ($F(8, 135) = 0.235$, $p = 0.964$). These facts implied that the error rates of the tested techniques were not affected much by the element size or layout density in the remote selection scenario.

We also conducted Dunnett tests as multiple comparisons in this study. The results are shown in Table 2. Combined with Figure 7, again we conclude that *2D-Dragger* was never worse than the other tested techniques. The details are as follows:

- *MagStick* had achieved similar performance with *2D-Dragger* in terms of completion time. However, its error rates were significantly higher than *2D-Dragger* for all configurations. This is mainly because the selectable region with *MagStick* did not cover the whole canvas, and it was thus impossible for the participants to select the target elements for some tasks. This fact suggests that although *MagStick* can double the selectable region, it might not be sufficient for some remote selection scenarios, such as selection on wall-size touch screen.

- *ObjectPointing* was still significantly slower than *2D-Dragger* when the elements were small. In this case, the participants needed to move the finger more carefully to avoid undesired switching of the captured elements, thus increasing the time of operation. In all density configurations, *2D-Dragger* was significantly faster than *ObjectPointing*. The reason was that, in each density configuration, there were tasks with small elements. Thus its overall performance was impeded by these tasks. In terms of the error rate, there was no significant difference between *2D-Dragger* and *ObjectPointing*.

- In this study, *Bubble+Touchpad* had achieved similar performance as *2D-Dragger* in terms of both completion time and error rate.

Summary

In both studies, the results showed that our tool outperformed the other techniques. Specifically, in Study I (reachable selection scenario), *2D-Dragger* and *MagStick* outperformed the other techniques in terms of both selection time and/or error rate. In Study II (remote selection scenario), *2D-Dragger* and *Bubble+Touchpad* were superior to the other techniques in terms of selection time and/or error rate. *MagStick* cannot reach some distant objects due to its reachability problem. Therefore these two studies confirmed that, as a unified tool, *2D-Dragger* was more powerful than the tested tools when dealing with both reachable and remote selection scenarios.

APPLICATIONS

Our method can be used in various application scenarios, and provides consistent experience on touch screen devices with different sizes. For example, our technique can benefit the following:

- General object selection with irregular layouts and sizes, such as text and hyperlinks in webpages, characters or objects in computer games, elements and controls in WYSIWYG editors (webpage, presentation slide, rich text document), etc. These scenarios require very accurate selection operation.

- Object browsing in applications like map location, photo wall, style effect preview in design software, etc. They usually need regular switching for information preview.

CONCLUSION

This paper has introduced *2D-Dragger*, a unified touch-based target acquisition tool, which is effective for different application scenarios and target layouts. Our main contributions are the design of the algorithm that makes universal selection possible without losing performance, and the user studies. No previous touch-based selection methods exhibit the desirable properties of *2D-Dragger*, namely, separated motion and display space, discrete selection, uniform effective width, no occlusion problem, and minimized finger movement. Our experiments show that *2D-Dragger* outperformed the previous tested selection tools both for reachable selection and remote selection.

LIMITATIONS AND FUTURE WORK

One limitation of our method is that the selection of void space (i.e., the space between elements) is not naturally supported by our technique. Another limitation is that when the screen is large and elements are distributed very densely, our method would not be helpful in selecting distant elements, because the finger needs to move a very long distance to select it. A possible solution is to employ the movement speed of the finger to accelerate target switching in unimportant regions, where the finger typically moves very fast. Another possible solution is to employ multi-touch operations. This will be explored in the future.

Acknowledgment

We thank all the reviewers of this paper for the insightful comments and constructive suggestions. This work was supported in parts by grants from Research Grants Council of HKSAR (16209514), City University of Hong Kong (7004298,7004498), NSFC (61232011), 973 Program (2015CB352501), Guangdong Science and Technology Program (2015A030312015, 2014B050502009) and Shenzhen Innovation Program (JCYJ20151015151249564).

REFERENCES

1. Pär-Anders Albinsson and Shumin Zhai. 2003. High precision touch screen interaction. In *CHI '03*. 105–112.
2. Oscar Kin-Chung Au, Xiaojun Su, and Rynson WH Lau. 2014. LinearDragger: a linear selector for one-finger target acquisition. In *CHI '14*. 2607–2616.

3. P. Baudisch, E. Cutrell, D. Robbins, M. Czerwinski, P. Tandler, B. Bederson, and A. Zierlinger. 2003. Drag-and-pop and drag-and-pick: Techniques for accessing remote screen content on touch-and pen-operated systems. In *Proceedings of INTERACT*, Vol. 3. 57–64.
4. Patrick Baudisch, Alexander Zotov, Edward Cutrell, and Ken Hinckley. 2008. Starburst: a target expansion algorithm for non-uniform target distributions. In *Proceedings of the working conference on Advanced visual interfaces*. ACM, 129–137.
5. Anastasia Bezerianos and Ravin Balakrishnan. 2005. The vacuum: facilitating the manipulation of distant objects. In *CHI '05*. 361–370.
6. Xiaojun Bi and Shumin Zhai. 2013. Bayesian Touch: A Statistical Criterion of Target Selection with Finger Touch. In *UIST '13*. 51–60.
7. Renaud Blanch, Yves Guiard, and Michel Beaudouin-Lafon. 2004. Semantic pointing: improving target acquisition with control-display ratio adaptation. In *CHI '04*. 519–526.
8. Andrew Bragdon and Hsu-Sheng Ko. 2011. Gesture Select:: Acquiring Remote Targets on Large Displays Without Pointing. In *CHI '11*. 187–196.
9. Olivier Chapuis, Jean-Baptiste Labrune, and Emmanuel Pietriga. 2009. DynaSpot: speed-dependent area cursor. In *CHI '09*. 1391–1400.
10. Leah Findlater, Alex Jansen, Kristen Shinohara, Morgan Dixon, Peter Kamb, Joshua Rakita, and Jacob O Wobbrock. 2010. Enhanced area cursors: reducing fine pointing demands for people with motor impairments. In *UIST '10*. 153–162.
11. Clifton Forlines, Daniel Vogel, and Ravin Balakrishnan. 2006. HybridPointing: fluid switching between absolute and relative pointing with a direct input device. In *UIST '06*. 211–220.
12. Tovi Grossman and Ravin Balakrishnan. 2005. The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. In *CHI '05*. 281–290.
13. Yves Guiard, Renaud Blanch, and Michel Beaudouin-Lafon. 2004. Object pointing: a complement to bitmap pointing in GUIs. In *Proceedings of Graphics Interface 2004*. 9–16.
14. Amy K Karlson and Benjamin B Bederson. 2007. ThumbSpace: generalized one-handed input for touchscreen-based mobile devices. In *Human-Computer Interaction–INTERACT 2007*. Springer, 324–338.
15. Wing Ho Andy Li and Hongbo Fu. 2013. BezelCursor: bezel-initiated cursor for one-handed target acquisition on mobile touch screens. In *SIGGRAPH Asia 2013 Symposium on Mobile Graphics and Interactive Applications*. ACM, 36.
16. Wing Ho Andy Li, Hongbo Fu, and Kening Zhu. 2016. BezelCursor: Bezel-initiated cursor for one-handed target acquisition on mobile touch screens. *International Journal of Mobile Human Computer Interaction* 8, 1 (2016).
17. David C. McCallum and Pourang Irani. 2009. ARC-Pad: Absolute+Relative Cursor Positioning for Large Displays with a Mobile Touchscreen. In *UIST '09*. 153–156.
18. Mathieu Nancel, Olivier Chapuis, Emmanuel Pietriga, Xing-Dong Yang, Pourang P. Irani, and Michel Beaudouin-Lafon. 2013. High-precision Pointing on Large Wall Displays Using Small Handheld Devices. In *CHI '13*. 831–840.
19. Alex Olwal, Steven Feiner, and Susanna Heyman. 2008. Rubbing and tapping for precise and rapid selection on touch-screen displays. In *CHI '08*. 295–304.
20. J Karen Parker, Regan L Mandryk, Michael N Nunes, and Kori M Inkpen. 2005. TractorBeam selection aids: improving target acquisition for pointing input on tabletop displays. In *Human-Computer Interaction-INTERACT 2005*. Springer, 80–93.
21. Richard L Potter, Linda J Weldon, and Ben Shneiderman. 1988. Improving the accuracy of touch screens: an experimental evaluation of three strategies. In *CHI '88*. 27–32.
22. Anne Roudaut, Stéphane Huot, and Eric Lecolinet. 2008. TapTap and MagStick: improving one-handed target acquisition on small touch-screens. In *Proceedings of the working conference on Advanced visual interfaces*. ACM, 146–153.
23. Xiaojun Su, Oscar Kin-Chung Au, and Rynson WH Lau. 2014. The implicit fan cursor: a velocity dependent area cursor. In *CHI '14*. 753–762.
24. Daniel Vogel and Patrick Baudisch. 2007. Shift: a technique for operating pen-based interfaces using touch. In *CHI '07*. 657–666.
25. Katrin Wolf and Niels Henze. 2014. Comparing pointing techniques for grasping hands on tablets. In *Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services*. ACM, 53–62.
26. Aileen Worden, Nef Walker, Krishna Bharat, and Scott Hudson. 1997. Making computers easier for older adults to use: area cursors and sticky icons. In *CHI '97*. 266–271.
27. Koji Yatani, Kurt Partridge, Marshall Bern, and Mark W Newman. 2008. Escape: a target selection technique using visually-cued gestures. In *CHI '08*. 285–294.