Touch Crossing-Based Selection and the Pin-and-Cross Technique

by

Yuexing Corona Luo

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2015

Abstract

This thesis focuses on the evaluation, exploration and demonstration of crossing paradigm with touch modalities. Under the scenario of crossing selection, the target is selected by stroking through a boundary "goal" instead of tapping or clicking on a rectangular graphical target. Crossing has been widely studied on stylus and mouse input, however, there is no generalizable, controlled empirical support for the application of crossing to touch input. In this work we present fundamental performance results for crossing-based selection tasks with direct touch input. Our study reveals that: touch crossing is faster than or equivalent to touch pointing; continuous selection of large orthogonal crossing targets is the most effective, and continuous selection of small collinear targets is least effective.

We model touch crossing with Fitts' law in order to characterize its performance. Unlike indirect stylus and mouse crossing, not every kind of direct touch pointing performance is modeled accurately with the standard Fitts' law. Instead, "FFitts" law, used previously for touch pointing with small targets, is employed to more accurately model discrete touch crossing with a directionally constrained target. In addition, visual touch feedback is shown to have a profound effect on task accuracy.

With multi-touch input, crossing semantics could become even richer than with pen or mouse. Inspired by the positive results from our study on touch crossing selection, we develop, evaluate and demonstrate a new multi-touch interaction space called "pin-and-cross", where one or more static touches ("pins") are combined with a cross stroke on a radial target to complete a selection, and all performed with one hand. A formative study and a comparative study are used to evaluate pin-and-cross, and an Android photo app is created to demonstrate four potential usages of pin-and-cross.

The formative study is conducted to reveal the kinematic characteristics of pin-and-cross, and to evaluate its fundamental performance, as well as subjective preference for target angles. These results are used to form design guidelines and a set of recognition heuristics for pin-and-cross menus invoked with one and two pin fingers on first touch or after a drag.

The controlled experiment compares a one-finger pin-and-cross contextual menu with a functionally equivalent Marking Menu and a partial Pie Menu. Results of this study suggest that pin-and-cross is just as accurate and even 27% faster than existed techniques when invoked on a draggable object.

Guidelines produced from the formative study are also used in the implementation of the photo app, which is created to demonstrate more pin-and-cross variations such as two-finger accelerated scrolling, drawing modes or attributes changing while drawing, two-finger constrained transformations, and pin-and-cross combined with a Marking Menu.

Lastly, with touch crossing being empirically validated as a practical and efficient selection technique, and pin-and-cross being demonstrated an faster menu selection tool, we hope that the results from this study will further inspire the exploration of novel forms of expressive multi-touch crossing techniques in the future.

Acknowledgements

I have created this thesis with the support of many people. In particular, I would like to thank:

- My supervisor, Daniel Vogel. Dan's creativity, intellect, guidance and encouragement inspired me to pursue research.
- The members of my committee: Edward Lank and Craig Kaplan. I really appreciate your time and help on polishing my research into this thesis.
- All my friends who ever helped, supported, and humored me.

Dedication

To my dear parents and to Gelin,
“My thesis is done, we can probably talk about the grad trip”

To myself,
“You made it!”

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Problems and Challenges

With the advances of Human-Computer Interactions in Post-PC era, direct input modalities such as stylus and touch have been brought to user for delivering amazing new experiences. Traditional "point-and-click" **WIMP GUIs** (Graphic User Interfaces based on Windows, Icons, Menus, and Pointers), which have been dominant for the past decades, are facing a considerable number of problems since they were originally designed for conventional desktop tasks.

In the desktop configuration, users select a target by pointing the cursor inside a perimeter and then clicking using the mouse. Hand occlusion can be avoided in this case since mouse uses indirect pointing technique where physical movement will not cover display. Precise selection can be achieved with mouse even on smaller targets such as the square icon to minimize the window. Right-click, double-click, middle-click on a three-button mouse are used to trigger more complex tasks, and keyboard shortcuts allow user to access command more efficiently. However, those benefits would reduce dramatically if we move from desktop to mobile devices and replace mouse with direct stylus or touch input, but stick on the "point-and-click" WIMP GUIs.

For example, compared with mouse clicking, a single tapping (Figure 1.1a), which can be seen as the replacement of mouse clicking in "point-and-click" interface, i.e., button-based interface, on stylus-driven devices or direct touch input devices, demands more physical space and efforts to accomplish. Double tapping requires even more. For both stylus and touch input, accuracy would further degrade if we are targeting on a smaller target [13, 19]. Efficient command invocation is also a problem [5] since it is unclear how to directly apply the concept of keyboard shortcut on stylus-driven or direct-touch devices. In conclusion, we need to find a more efficient, expressive and flexible interaction technique to bridge the gap between advanced input modalities and conventional interfaces.

One possible starting point is to introduce new target selection techniques. Intuitively, stylus is considered as a tool of drawing. It would be more natural if we make selections by drawing instead of tapping or clicking when we place a stylus on a flat surface. Inspired by this, *crossing paradigm* (Figure 1.1b), where the target is selected by stroking through a boundary “goal”, has been proposed and investigated [3][19] on stylus-driven devices, with the goal of bringing smoother interaction experiences to users.

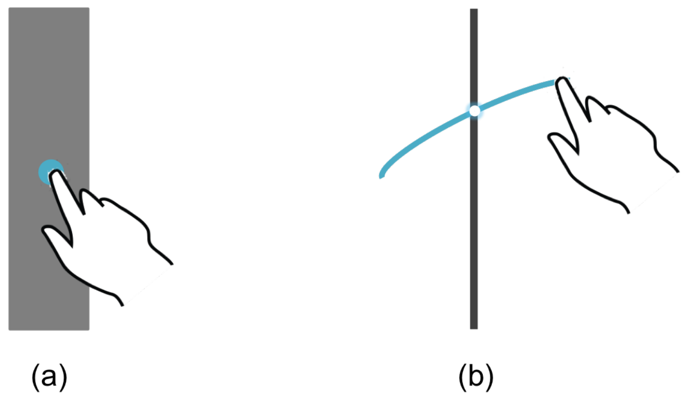


Figure 1.1: (a) Tap on a rectangular target to select; (b) Cross though a boundary “goal” to select

With the increasing popularity of direct touch input, the crossing paradigm has also been applied to touch screen techniques [11, 41, 43, 46, 54]. However, unlike stylus input, there is no generalizable and controlled empirical support for the application of crossing to touch input. Most of the existing work of touch crossing actually rely on the evidence provided for stylus crossing. Studies of conventional touch input have noted unique characteristics compared to other input [7, 10, 20, 28], some of which might affect the crossing performance. Thus it is a bit problematic to design crossing interfaces with touch input but following guidelines generated from stylus input.

Another major motivation for validating touch crossing as a practical and efficient selection technique, is to explore other novel forms of touch crossing, to further expand its expressibility, and ultimately to achieve efficient command invocation as mentioned before. One possible solution is to combine continuous crossing with discrete touch input, or what’s more exciting, with multi-touch input. For example, we can keep one finger stationary on the touch screen to activate crossing targets for neighboring fingers to cross. This could pack more crossing targets within a small area, and only activate them when a standard touch event is detected without resorting to dwell time. We will return to this idea in detail later in this chapter.

In sum, combining touch input with crossing may provide a way to further increase touch expressibility and help close the performance gap when abandoning mouse and keyboard. However, before we unlock more potentials, we need empirically validated evidence to support the applicability of touch crossing interfaces. We will discuss both stylus and touch input. However given the increasing popularity of direct touch devices, we will pay more attention on touch modality.

1.2 Contributions

Our work has two major contributions: 1) a set of empirical evidence and theoretical results to provide fundamental support for crossing-based interaction techniques in direct touch interfaces (Chapter 2); 2) a new technique called *pin-and-cross* where one or more static touches (“pins”) are combined with crossing interfaces to further expand its design space (Chapter 3-5). Both of these contributions will now be examined in detail.

1.2.1 Touch Crossing Selection

We present an in-depth investigation of touch crossing selection tasks. Our work presents a set of new controlled empirical evidence and theoretical results that provide fundamental support for crossing-based interaction techniques in direct touch interfaces. Specifically, we adapt Accot and Zhai’s indirect stylus crossing experiment [3] to direct touch input, and find that, with directionally constrained targets, continuous touch crossing is faster than touch pointing, and discrete crossing performs similarly to pointing. Our results suggest a minimum crossing target width of 12.7mm will reduce error rates to 4% or less. We also confirm a trend observed for stylus input: the selection of large directionally constrained crossing targets is most effective and continuous selection of small collinear targets is least effective. For a continuous crossing task with direction constraint, direct touch is approximately 37% faster than indirect stylus, and 58% faster than direct stylus crossing in a similar study by Forlines and Balakrishnan [19].

We also found that standard Fitts’ law accurately models all crossing tasks except discrete crossing with a direction constraint. To address this, we utilize “FFitts law” [13], a formulation of Fitts’ law for finger touch tapping input which takes into account the contribution of absolute touch precision independent of the speed-accuracy tradeoff effect. We conducted the second experiment to measure absolute touch precision with crossing and demonstrate the applicability of FFitts law for improved modeling of discrete crossing with a direction constraint. Continuous crossing with a direction constraint is modeled equally well with FFitts law, though not as well as with Fitts’ law. Our second experiment also controls for visual touch feedback and demonstrates its pronounced affect on absolute touch precision.

1.2.2 Pin-and-Cross

With touch crossing performances being validated, now we are able to further explore other novel forms of crossing interfaces. In Chapter 3 we introduce pin-and-cross, where static touches ("pins") are combined with crossing. With a pin-and-cross context menu, for example, one finger pins an object and neighboring finger crosses a line target to select a command (Figure 1.2). Targets can be crossed in either direction for different commands. Since the pin finger does not move, the object can still be dragged without entering special mode. Moreover, we can use two or even more fingers to further expand the input space of "pin". We focus on unimanual pin-and-cross techniques using one or two pin fingers with radial targets, and combine it with scrolling, dragging, and transformation, to carry out space-conserving mode selection or command invocation.

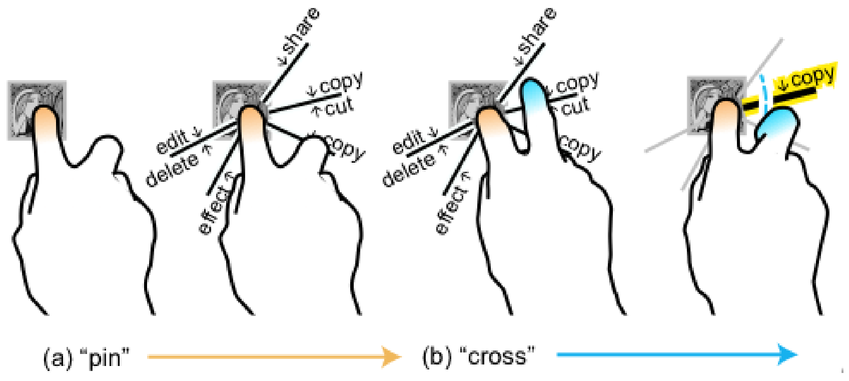


Figure 1.2: Unimanual *pin-and-cross* context menu: (a) one finger pins an object; (b) another finger crosses a radial target.

Specifically, we contribute a systematic exploration of the pin-and-cross touch input space, and generate a set of design guidelines for pin-and-cross techniques with empirically derived heuristics to recognize them. We also conduct a comparison experiment to validate pin-and-cross against functionally equivalent Marking Menu [38] and partial Pie Menu [8], and prove that pin-and-cross is just as accurate and 27% faster when invoked on a draggable object. Additionally, a tablet photo app is presented demonstrating four pin-and-cross interaction techniques.

1.3 Outline

This chapter has described the problems of conventional button-based interfaces when abandoning mouse and keyboard, provided motivations for touch crossing as an interaction technique and introduced a new multi-touch crossing technique called pin-and-cross.

The rest of this thesis is organized as follows: Chapter 2 will report the results of an empirical study that fundamentally validates the performance of touch crossing. In Chapter 3 we will explore pin-and-cross with a formative study and formulate design guidelines for pin-and-cross interfaces. Chapter 4 is a comparison study where we evaluate pin-and-cross against functionally equivalent marking menu and pie menu. Chapter 5 will introduce a heuristic-based recognizer for pin-and-cross, along with an a set of demonstration applications to prove its applicability. In the last chapter we will summarize the entire thesis and propose potential future work.

Chapter 2

Empirical Evaluation of Touch Crossing-Based Selection

In this chapter we present fundamental performance results for crossing-based selelction tasks with direct touch input. First we describe previous research related to crossing-based selection and identify the research issues addressed in this work. Next, we expand our investigation by taking a close adaptation of Accot and Zhai's [3] indirect stylus crossing experiment, which reveals similar trends for direct touch input: touch crossing task time is faster or equivalent to touch pointing; continuous selection of large orthogonal crossing targets is most effective; and continuous selection of small collinear targets is least effective.

We also find that standard Fitts' law is able to accurately model all crossing tasks except discrete crossing with a direction constraint. Thus we extend our study of modeling touch crossing by utilizing "FFitts Law" [13], a formulation of Fitts' law for finger touch tapping input which takes into account the contribution of absolute touch precision independent of the speed-accuracy trade-off effect. We conduct a second experiment to measure absolute touch precision with crossing and demonstrate the applicability of FFitts law for improved modeling of discrete crossing with a direction constraint. As a result, continuous crossing with a direction constraint is modeled equally well with FFitts law, though not as well as with Fitts' law. Our second experiment also controls for visual touch feedback and demonstrates its pronounced effect on absolute touch precision.

Finally, the results from two experiments are interpreted to provide design rationale behind touch crossing interfaces.

2.1 Related Work

As we discussed before, though crossing interfaces have been implemented on various applications including touch modalities, almost all empirical evaluations of elementary crossing performance use stylus or mouse input. Techniques using stylus crossing are easily justified with existing evidence showing that crossing with a stylus can be faster and less error prone than pointing [3, 19]. Conversely, a well controlled study found that crossing with a mouse is sub-optimal compared to pointing [51]. In spite of the ubiquity of mouse input, this could explain why few applications use mouse crossing. However, despite the wide usage of touch crossing techniques, there is no controlled and generalizable empirical evidence to justify touch crossing.

2.1.1 Crossing-Based Selection with Stylus

Accot and Zhai's performed an early crossing-based selection experiment which they referred to as a "goal passing task" [1]. This formed the first step in the development of their Steering Law, but also led to their subsequent examination on crossing [2]. In this thorough crossing study, Accot and Zhai compared crossing to pointing with indirect stylus input under six different task conditions. Figure 2.2 shows their most cited model. Their experiment also reveals that for a goal-crossing task, the time $\boldsymbol{T}$ to cross a goal can be determined by the goal width $\boldsymbol{w}$ and goal distance $\boldsymbol{d}$. Formally, the relationship among $\boldsymbol{T}$, $\boldsymbol{w}$, $\boldsymbol{d}$ is addressed in the following equation:

$$T = a + b \times \log(\frac{d}{w} + 1) \tag{2.1}$$

This equation takes the same form as in Fitts' Law. The logarithmic factor in Equation 2.1 is referred to as the *index of difficulty (ID)* of pointing or crossing task. ID is used to describe the difficulty of motor tasks.

Note that we use the same six tasks in our study with direct touch input (Figure 2.2), but name them with the refined task terminology of Apitz et al. [5]. In the Accot and Zhai study and our study, two tasks require pointing selection and four require crossing selection. For both kinds of selection, the orientation of the precision constraint for a goal or target can be collinear to movement, so that controlling movement amplitude is most important, or it can be orthogonal to movement, so that controlling movement direction becomes the most important. When crossing, the continuity of contact is continuous if the stylus remains on the surface between targets, or discrete when it must lift up to avoid distractors.

With indirect stylus input, Accot and Zhai find that pointing and continuous crossing with a direction precision constraint (*DP* and *C/DC*) are at least 10% faster than all other tasks. In contrast, continuous crossing with an amplitude precision constraint (*C/AC*) is slower than all other tasks. All tasks except *C/AC* have error rates close to, or lower, than 7.8%, the error rate for pointing. In addition, Fitts' law model all tasks extremely well (all r^2 greater than 0.975). Their study provides evidence that crossing can be substituted for pointing with no decrease in performance, however this can only be confidently stated for an indirect stylus device with absolute mapping.

Forlines and Balakrishnan [19] also use Accot and Zhai's six tasks to study direct and indirect stylus input with tactile feedback. They report no difference in task time between direct and indirect stylus input when pointing, but when crossing, direct stylus input is 16% faster. They also report a surprising interaction between input device and continuity of contact. Discrete crossing tasks are 11% faster with direct stylus input, but 15% slower with indirect stylus input. This nuanced performance emphasizes the need to test and validate crossing performance across input modalities. Perhaps due to the inclusion of a small 0.7mm target, they report a mean error rate of 20% for crossing, much higher than Accot and Zhai's work. Additionally, Forlines and Balakrishnan find that that crossing tasks can be further enhanced on direct input with tactile feedback. This inspires us to include visual touch feedback as one controllable variable that may affect crossing performance in our second experiment.

Apitz et al. [5] illustrate several applications with crossing-based interfaces, and study parameters that influence the performance of crossing, like the landing and take-off space of input stylus, and the angle between the target centerline and the horizontal line. Motivated by potential benefits of crossing selection, Dixon et al. [16] test crossing target density and orientation in parameter selection dialog boxes. Using a direct stylus input device they find crossing is faster than pointing for dialog boxes, and also remains spatially efficient. Apitz and Guimbretiere's CrossY [4] is a fully crossing-based drawing application using direct stylus input. It illustrates many crossing-based widgets and interaction techniques. Although no formal study was conduced with CrossY, the empirical evaluations of elementary crossing performance with stylus justify the design.

2.1.2 Crossing-Based Selection with Mouse

Using a mouse and trackball, Wobbrock and Gajos [51] compare pointing and crossing with able-bodied people and those with motor impairments. They conclude that able-bodies users are better with area pointing than crossing, while users with motor impairment are better with crossing selection. Although not the primary focus of their work, the data from able-bodied participants in 15 controlled combinations of distances and widths provide evidence that crossing with a mouse is sub-optimal. Feng et al. [18] propose a selection method called *target reverse crossing* which requires people control the mouse pointer by first entering the target region and then leaving it by crossing a certain target edge. This approach is also designed specifically for people with motion impairments and used under a camera-based system. They conduct an experiment to compare the performance of reverse crossing selection and *dwell time* selection, which requires maintaining the pointer in the target region for a certain period of time to achieve command selection. However, the comparison results are debatable. In the experiment, Feng et al. set up the dwell time as 1 second, which doubles the common dwell time of around 0.5s [12] in camera-based system. General "point-and-click" interfaces require even shorter dwell time (0.33s) [35].

Another example of combining crossing with timing are Dragicevic's Fold-and-Drop [17], where boundary crossing is used to manage overlapping windows. They propose *timed double-crossing*, and state "*timed double-crossing can be seen as a crossing-based equivalence to double-clicking*". They address the benefits of mouse crossing in the context of dragging tasks: pointing-based tasks are not performable during the time of a drag (unless using other buttons which cost extra space and effort). Additionally, mouse crossing can be used to control a set of widgets simultaneously. Baudisch's Toggle Maps [X] allows the manipulation of several toggle switches with a single mouse crossing interaction. Perin et al. [42] design *crossets* interface to manipulate multiple sliders at once. This technique is employed to optimize interactive tabular visualizations.

2.1.3 Crossing-Based Selection with Touch

Various crossing-based applications and interaction techniques have been proposed and implemented with touch input, in spite of the absence of controlled and generalizable empirical evidence to justify them. Benko et al. design a crossing-based menu for a multi-touch tabletop [11]. Sulaiman and Olivier's Attribute Gates [46] use crossing for attribute selection. Roth and Turner's Bezel Swipe [43] is crossing with one side of a smartphone bezel as the goal. Yoshikawa et al.'s HandyWidgets [54] create implicit crossing goals defined by hand position on a multi-touch tabletop. Nakamura et al. [41] create a new interaction technique called double-crossing and claim improvements on both selection time and error rate.

These examples of touch crossing are exciting, but to our knowledge, comparisons of touch pointing and crossing appear only twice. Kay et al. [32] compare the execution time across four touchscreen input techniques including touch down, goal crossing, finger lift, finger tilt and physical button in their PVT-Touch study with 20 subjects. However, the objective of their study is to address the benefits of touch over physical buttons. In their experiment, action can be activated anywhere on the screen of an Android phone, and they split a standard touch into touch down and finger lift and report that goal crossing has higher execution time than either of them. Another is Guerreiro et al.'s study with 15 paraplegic participants using a smartphone [22]. They use single, circular crossing targets larger than 7 mm and did not control for distance. Due to impairment, participants do not necessarily use a finger to touch. The paper reports tapping and crossing performance as similar, which is encouraging. However, the study lacks experimental control for key factors used in Accot and Zhai's work, and the exclusive use of motor impaired participants makes generalizing these results beyond their intended purpose problematic.

Existing works on touch crossing justify themselves by referring evidences generated for direct stylus crossing, but investigations into properties of touch reveal many subtle and not-so-subtle differences. The fat finger problem is one pronounced difference [48]. The centroid touch point model is not sufficient if high accuracy is required [27] unless we use advanced fingerprint sensors [28]. The problem with friction between fingers and the surface when dragging has been examined on both Cockburn et al.'s study on direct touch [15], and Buxton's study of indirect touch [14]. This problem is most likely to influence the performance of touch crossing. Benko and Wigdor's overview of touch input problems [10] enumerates issues with precision, occlusion, capture, and physical constraints.

So far, the extensive empirical studies available to guide and justify touch pointing do not exist for designers and researchers exploring touch crossing as an alternative or complementary interaction technique. To remedy this situation, we conduct a controlled experiment to explore and validate touch crossing performance and characteristics.

2.2 Experiment 1: Touch Crossing Performance

This experiment is a very close adaptation of Accot and Zhai's indirect stylus crossing experiment [5, 3], which has also been used by Forlines and Balakrishnan [19] for direct stylus input. It is hoped that by closely following an established design, the validity of the experiment is stronger and direct comparisons with previous results for direct and indirect stylus are possible. However, one difficulty for such comparison is inconsistent measurement. Accot and Zhai, Forlines and Balakrishnan use distances and widths in pixel, which may cause inaccuracy during the comparison when switching devices. Not mention that Accot and Zhai use an indirect pen on an absolutely-mapped tablet with dimensions 63% smaller than the display. Though same set of IDs can be generated from the combination of target width and distance applied in previous experiments, the real distance travelled in motor space differs. This will affect the value of $\boldsymbol{a}$, $\boldsymbol{b}$, and the overall time performance afterwards in Equation (2.1). We will talk about how we tackle this issue in Section 2.2.1.

Our adaption of Accot and Zhai's experiment is described in detail below, the main changes are:

- All target distances and sizes are scaled by a ratio so the maximum distance would fit within a typical touch tablet. By using a ratio, task IDs remain unchanged.
- The largest target width is removed and a smaller width inserted to focus our evaluation on relatively high ID tasks with practical dimensions.

2.2.1 Experiment Design

Tasks and Stimuli

The primary independent variable is *TASK* type. We use the same six tasks as Accot and Zhai, but we follow the revised terminology introduced in Apitz et. al. [5]. Three types of variables are identified within these six conditions to evaluate the differences between pointing and crossing, as well as factors that affect the performance of goal crossing. More precisely, we evaluate both pointing and crossing task where pointing is seen as the baseline.

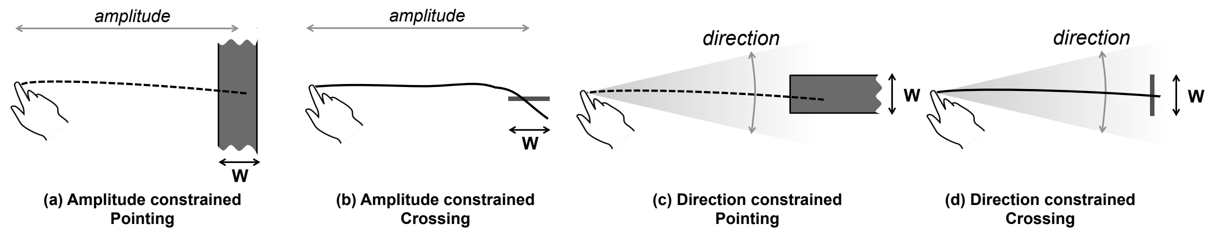

amplitude
W
(a) Amplitude constrained Pointing
amplitude
W
(b) Amplitude constrained Crossing
direction
W
(c) Direction constrained Pointing
direction
W
(d) Direction constrained Crossing

Figure 2.1: *Amplitude*/*Direction* constrained *Pointing*/*Crossing* tasks.

Note that there are different ways to orient targets and to select crossing targets in sequence. For pointing, one dimensional Fitts' law task typically looks like Figure 2.1(a), where targets have a near infinite height, but relatively small width parallel to the movement amplitude. Such task has been named as *Amplitude Constrained Pointing* because the movement amplitude must be accurate to hit the target. In other words, to hit the target successfully, it is how far you move that matters. For crossing, the same amplitude constraint can be applied as shown in Figure 2.1(b). But this is not how we usually cross targets.

Intuitively, people prefer to cross targets that are facing the same direction they are moving, as illustrated in Figure 2.1(d), where the movement amplitude is perpendicular to the crossing target. This task is named as *direction constrained crossing* because the direction of movement determines whether target will be hit or not instead of amplitude. *direction constraint pointing*, as shown in Figure 2.1(c), has an target with infinite width parallel to the movement amplitude.

Unlike pointing, there are actually two different ways to select more than one crossing target in a certain sequence. With a continuous style, user select targets by keeping finger on the screen from the start to the end (Figure 2.2(e),(f)), where direction and amplitude still play an important role when approaching targets. Sequential selection can also be achieved with a discrete style where finger lifts up between targets as shown in Figure 2.2(c),(d). Note that in our experiment we add a barrier between targets to visually enforce user lifting hand. Direction and amplitude constrained tasks are both examined under discrete crossing style as well.

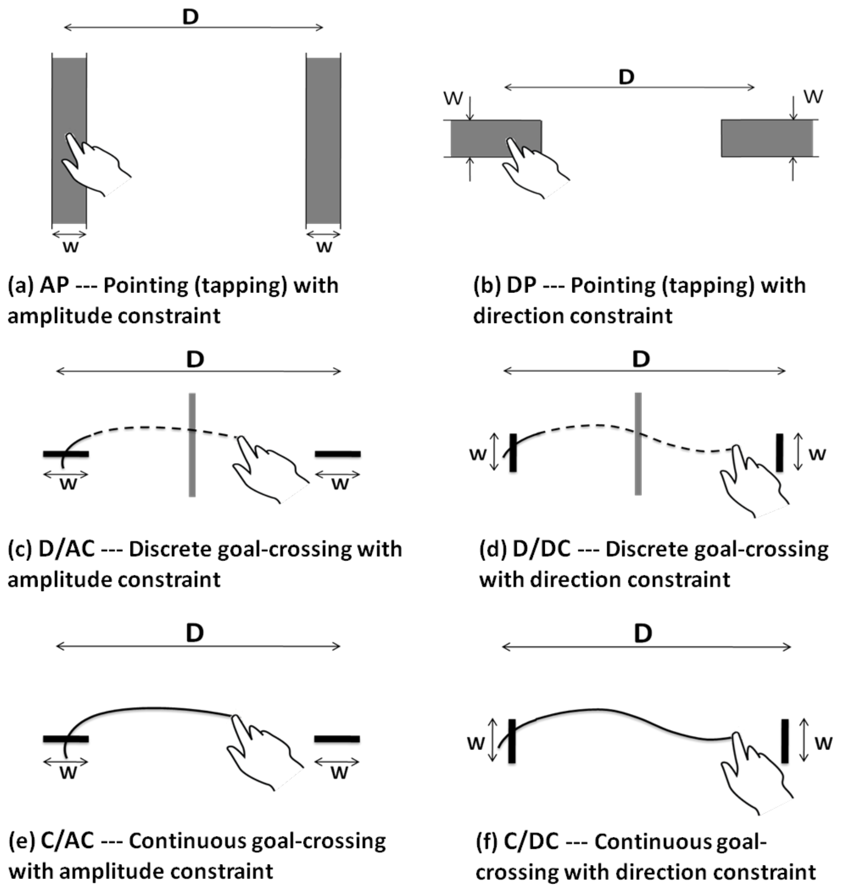

D
w
w
(a) AP --- Pointing (tapping) with amplitude constraint
D
W
W
(b) DP --- Pointing (tapping) with direction constraint
D
w
w
(c) D/AC --- Discrete goal-crossing with amplitude constraint
D
w
w
(d) D/DC --- Discrete goal-crossing with direction constraint
D
w
w
(e) C/AC --- Continuous goal-crossing with amplitude constraint
D
w
w
(f) C/DC --- Continuous goal-crossing with direction constraint

Figure 2.2: The six *TASK* conditions.

In summary, there are two reciprocal pointing tasks: *AP* and *DP*, and four reciprocal crossing tasks: *D/AC*, *D/DC*, *C/AC*, *C/DC*. Within these six tasks, we examine both pointing and crossing, different ways approach a target, as well as ways to select crossing targets in sequence. All *TASK*s are illustrated in Figure 2.2 and described in detail below:

AP: *Amplitude constrained pointing*. This is a traditional one-dimensional Fitts' tapping task [40]. The "infinite" vertical dimension of two rectangular targets imposes a precision constraint along the movement amplitude. In other words, since the smallest dimension of the rectangular target is parallel to the movement path, it requires the movement distance (called "amplitude" in Fitts' law studies) to be more precise than the movement direction. We refer to the size of the precision constraint as the target width *W*. The distance between the midpoints of the targets is *D*.

DP: *Directional constrained pointing*. Tapping is the selection action, but the rectangular targets are rotated to be "infinite" horizontally. This imposes a precision constraint W along the approach direction of the movement path. In other words, since the smallest target dimension is orthogonal to the movement path, it requires the movement direction to be more precise than the movement amplitude. The distance between targets is *D*.

D/AC: *Discrete, amplitude-constrained crossing*. Instead of pointing, the task is to cross two horizontal goals of width *W* with midpoints separated by distance *D*. Each goal is crossed with a downward stroke in a reciprocal pattern. A barrier between targets makes each goal crossing discrete (the finger must lift to pass over the barrier, or an error tone sounds).

D/DC: *Discrete, direction-constrained crossing*. Two vertical goals of width *W* are each crossed left-to-right in a reciprocal pattern. A barrier makes the task discrete.

C/AC: *Continuous, amplitude-constrained crossing*. The same task as *D/AC*, but no barrier and the finger must remain in continuous contact with the display throughout the reciprocal task, or else an error tone sounds.

C/DC: *Continuous, direction-constrained crossing*. The same task as *D/DC*, but with continuous finger contact throughout the reciprocal task.

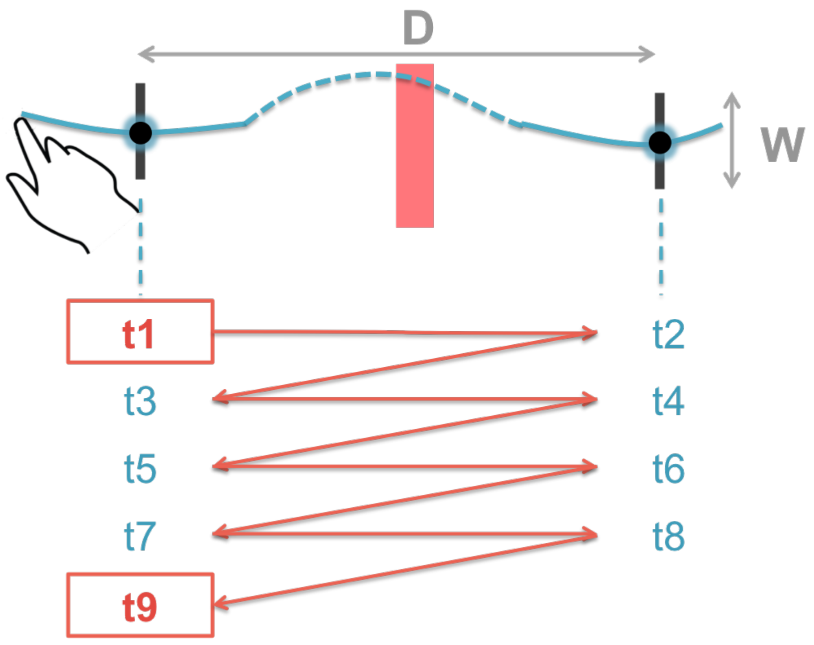


Figure 2.3: Average task (D/DC as an example) selection time.

Task Time and Errors

In our experiment, each task requires 9 target or goal selections, each selection alternating between right and left targets. The next target to select is shown in green and the other target orange. The time duration between two serial target selections creates one measurable data point, resulting in 8 measurable data trials as shown in Figure 2.3. We then calculate the mean value of these 8 measuable point to get the avg task selection time according to Equation 2.2.

$$\frac{1}{8}\sum_{i=1}^{8}(t_{i+1}-t_i) \tag{2.2}$$

Participants were instructed to perform all tasks as quickly and accurately as possible.

To keep participants focusing on accuracy, an error tone sounds when a target/goal is missed and that target/goal has to be completed before continuing. This produces extra tapping or crossing trials. We remove the error trials in the time analysis. A crossing is successful if the line segment formed by two successive touch points intersects with the line describing the crossing target. An error tone will sound when the same line segment intersects with the infinite line collinear with the crossing target.

Note that most tablet operating systems provide no cursor-like feedback for touch points, so we initially used no feedback as well. However, a 6-person pilot study with no feedback had very poor performance (a comparison to this pilot data is in our later discussion of section 2.2.3). Cursor feedback for touch has been shown to be important [25]. For the main experiment, we used simple cursor feedback in the form of a gray dot at the touch position with a radius 4.6mm transparent circle outlined in a 1px blue stroke (Figure 2.12 *FEEDBACK C*).

		Width						**Distance**	
Visual Space (px)		4	8	16	32	64	128	256	1024
Motor Space (mm)	Accot and Zhai	-	1.0	2.0	4.1	8.1	16.2	32.5	129.9
	Forlines and Balakrishnan	0.7	1.4	2.8	5.6	-	-	45.0	179.8

Table 2.1: *W* and *D* values used in previous studies

Distance and Width Combinations

As we discussed before, one of our objectives is to enable direct comparison between our touch crossing and previous work with stylus. Thus we choose values for independent variables *W* and *D* to approximately reproduce six out of seven task IDs tested by Accot and Zhai and four out of eight IDs tested by Forlines and Balakrishnan. We accomplish this by applying a scaling ratio to previous *W* and *D* values for applicability to tablets. The ratio, along with the values of *W* and *D* used in our experiment, are generated as described below.

First we constrain the maximum *D* value to be less than 203 mm, the display width of a 245 mm (10") 4:3 tablet. By using these additional distance constraint, we establish the largest distance as *D* = 203 mm (666px), which creates a scaling ratio of 0.65 given Accot and Zhai's largest distance of 1024px. The ratio is used to determine the shortest distance *D* = 50.8mm as well. This ratio assure us a wide coverage of IDs used in previous work, and produce measurements suitable for our tablet.

Distance (mm)	50.8	50.8	50.8	50.8	50.8	203.3	203.3	203.3	203.3	203.3
Width (mm)	3.2	5.0	6.4	12.7	25.4	3.2	5.0	6.4	12.7	25.4
ID (bits)	4.1	3.0	3.3	2.0	**1.0**	**6.0**	5.0	5.3	4.0	3.0

Table 2.2: *W*, *D* and IDs used in our experiment

However, we then find one problem in producing our own *W* values by directly applying this ratio. As shown in Table 2.1, Accot and Zhai's smallest 8px target width, translated from 1.0 mm in motor space on the 7.9 px/mm indirect stylus tablet they used, would scale to 1.6mm (5px) on our device. Forlines and Balakrishnan use an even smaller 0.7mm target. These target sizes are small even for stylus input, and are well below recommended touch target sizes [28]. Early pilots confirmed that very small crossing targets had high error rates, so we remove the smallest 1.6 mm. The remaining scaled target/goal widths are W = 3.2, 6.4, 12.7, and 25.4 mm. We insert a fifth W = 5.0 mm to provide additional ID data points and access to mobile-sized applications, since 5mm is the key width of the touchscreen keyboard on Galaxy Nexus. Across all D-W combinations, the ID range is 1.6 to 6.0. Table 2.2 shows our measurements.

Pilot Study of Crossing Landing/Taskoff Area

Consider that the maximal *D* must leave reasonable landing and takeoff space for crossing trajectories [16], we conduct a pilot experiment with 12 people, all right-handed to estimate the natural landing and takeoff space of crossing selection. Participants are asked to cross a single, centered 30.5mm vertical crossing goal 50 times from the left, then another 50 times from the right in rapid succession. We find a mean landing area of 290.86mm and mean takeoff area of 217.21mm. We admit that this is a highly unconstrained task, so these means are more representative of upper bounds. We report the values here for reference purposes only.

Participants

Twelve right-handed participants were recruited (8 female), ranging in age from 22 to 34. All reported experience with touch screen devices.

Apparatus

The experiment was performed on a 3M M2256PW capacitive touch display. The 513.2×337.8mm display has a resolution of 1680×1050px resolution creating an effective density of 3.3 px/mm (90 DPI). This display was selected due to its very low 6 ms latency. The display was laid on a standard table, tilted at an angle of 10° towards the participant. Participants sat in a standard office chair. Our experiment code is written in Java using the Multi-touch For Java (MT4j) library. It runs on a Phenom 9950 Quad-Core 2.6 GHz Processor with 8 GB ram. All touch input events and experiment events are logged at approximately 60 Hz with no noticeable user interface lag.

Design and Protocol

The experiment design is within-subjects, repeated measures, and full factorial. For each of the six *TASK*s, participants are asked to perform 1 full practice block and 2 consecutive measurement *BLOCK*s of 10 sets of D-W trials. Each set of trials covers one D-W combination and required 9 reciprocal selections of targets or goals to create 8 measured trials between successive selections. *TASK* order is counterbalanced using a Latin Square and the order for the set of D-W trials is randomized.

A short instruction and demonstration block is presented at the beginning of each *TASK* with three trials for one D-W combination (*D* = 121mm, *W* = 30mm, ID = 2.32). Short rest breaks are enforced at the end of each task and encouraged after each block. The experiment tasks 45 minutes on average. In summary:

6 *TASK*s (*AP*, *DP*, *D/AC*, *D/DC*, *C/AC*, *C/DC*) $\times$
2 *BLOCK*s $\times$
2 distances *D* (203.8, 50.8 mm) $\times$
5 widths *W* (3.2, 5.0, 6.4, 12.7, 25.4 mm) $\times$
8 measured target selections
= 960 data points per participant

2.2.2 Results

Learning Effect

Although the first block is intended to function as a practice block, we can still use it to test for leaning effects since it is a full block. Including the practice block, we found

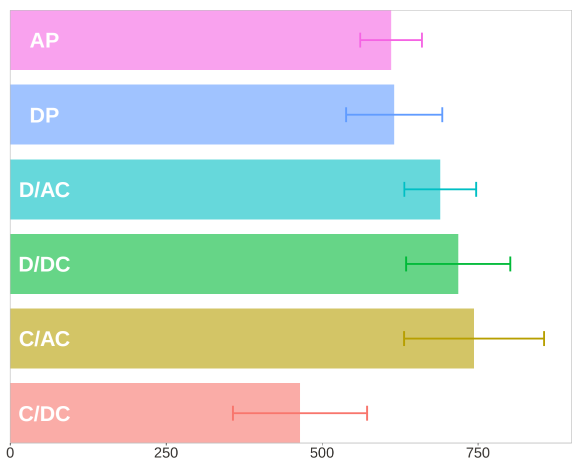


Figure 2.4: Selection time by *TASK*

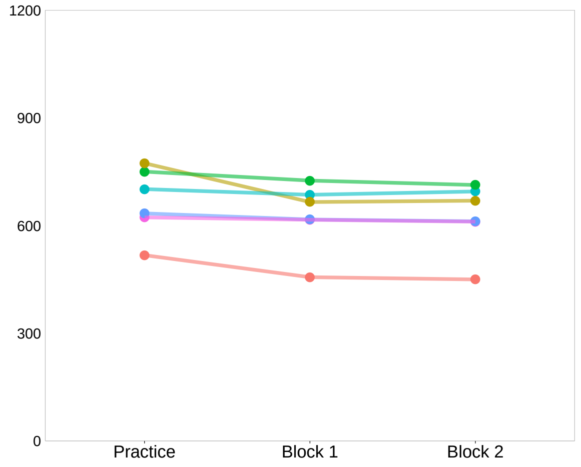


Figure 2.5: Effect of *BLOCK* on task time

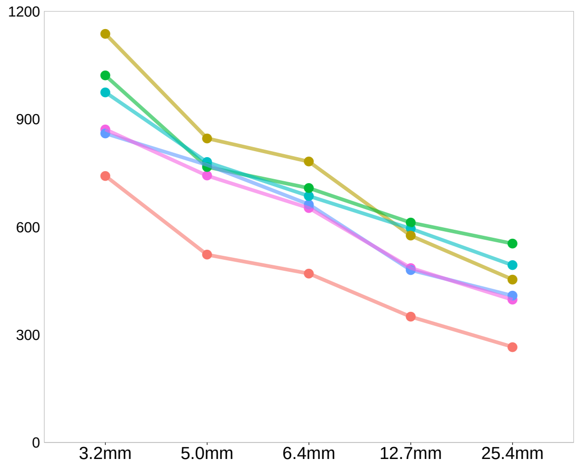


Figure 2.6: Effect of *W* on task time

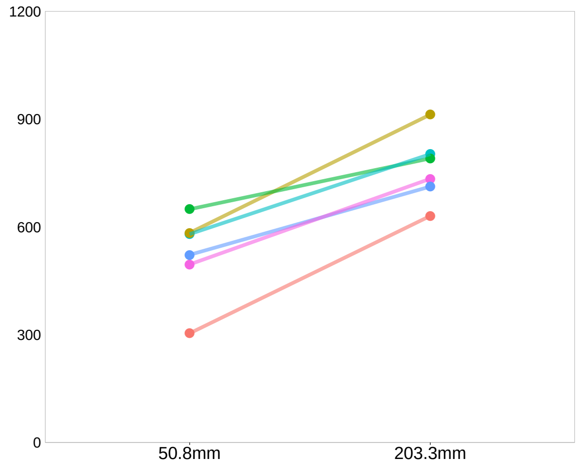


Figure 2.7: Effect of *D* on task time

no significant effects within *BLOCK* or for *BLOCK* $\times$ *TASK* (Figure 2.5), confirming no learning effects are presented in our experiment.

Selection Time

We define selection time as the mean duration of each measured selection. There is a main effect of *TASK* on selection time ($F_{5,55} = 13.91, p < 0.0001$). As illustrated in Figure 2.4 [1], the time performance of pointing and discrete crossing are similar. Post hoc analysis[2] shows that continuous crossing tasks are significantly different from pointing tasks (all $p < 0.01$), and that *C/DC* is the fastest (464ms, 190 STD) among all *TASK*s (all

[1]all error bars in figures are 95% CI

[2]All post hoc tests use the TukeyHSD adjustment.

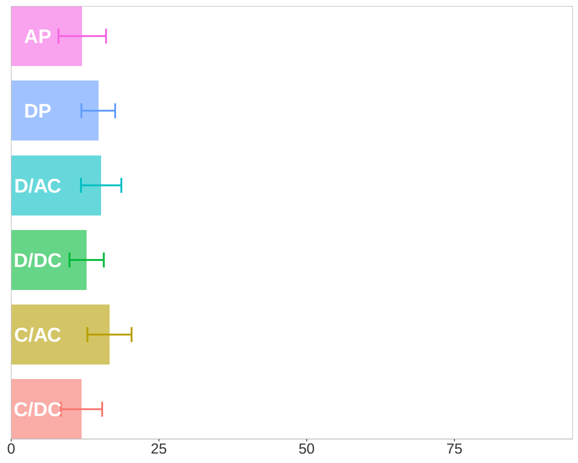


Figure 2.8: Error rate by *TASK*

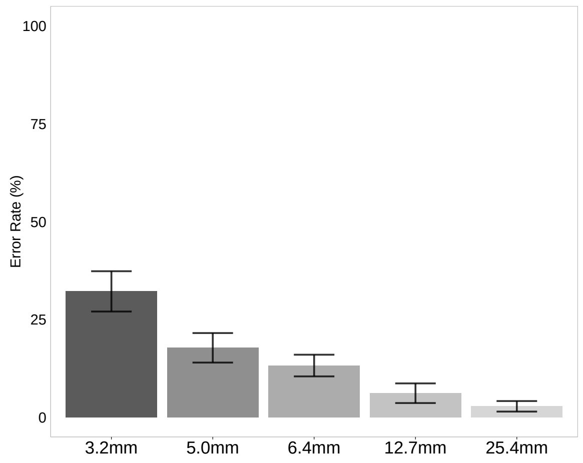


Figure 2.9: Effect of *W* on crossing errors

$p < 0.001$). Note that *C/AC* was slowest (743ms, 198 STD) suggesting that we might avoid the design of amplitude constrained crossing targets. There are no significant differences between *AP* (610ms, 87 STD), *DP* (615ms, 136 STD), *D/AC* (689ms, 198 STD), or *D/DC* (718ms, 148 STD).

As predicted by previous work, movement distance *D* significantly affect selection time ($F_{1,11} = 222.4, p < 0.0001$). The time to select targets at the long distance is 764 ms (188 STD), 46% slower than the short distance (Figure 2.7). There is also an interaction effect between target width *W* and time ($F_{4,44} = 113.5, p < 0.0001$). Post hoc analysis indicates that all widths are significantly different with each other (all $p < 0.04$), see Figure 2.6. The mean time for *W* = 3.2mm is 934 ms (260 STD), 59% slower than the mean time of all other widths, and more than twice the mean completion time of *W* = 25.4mm at 426 ms (120 STD).

Error Rate

There is a main effect of *TASK* on error ($F_{5,55} = 2.535, p = 0.039$). But given high probably *type I error*, post hoc analysis found no differences. Task error rates are similar (AP 12%, DP 15%, D/AC 15%, D/DC 13%, C/AC 17%, C/DC-12%). Our error rates are higher than Accot and Zhai, but lower than Forlines and Balakrishnan for all tasks except D/AC (see 2.2.3 and Figure 2.8). Note that our smallest *W* = 3.6 mm is small for touch input [18] and Forlines and Balakrishnan test extremely small targets (0.7mm), so higher error rates are expected.

Main effect has been found for *D* on error ($F_{1,11} = 34.94, p < 0.001$). Post hoc tests find error rates are marginally higher for D = 203.3mm (16%) compared to D = 50.8mm are (12%). We believe that the longer distance encourages a higher velocity which reduces crossing accuracy.

There is also a main effect of W on error ($F_{4,44} = 11.75, p < 0.0001$). Post hoc tests reveal that all pairs of widths are significantly different (all $p < 0.0001$), except for W=5.0mm with W=6.4mm, and W=12.7mm with W=25.4mm. The significance and quantity of errors for widths remains the same even if only crossing tasks are analyzed (Figure 2.9). Given that error increases from 4% for W=12.7mm to 16% for W=6.4mm, we recommend touch crossing targets no smaller than 12.7mm. We have not investigated it further, however current result suggests that targets no smaller than 12.7mm is more suitable for touch crossing based interfaces.

Preference

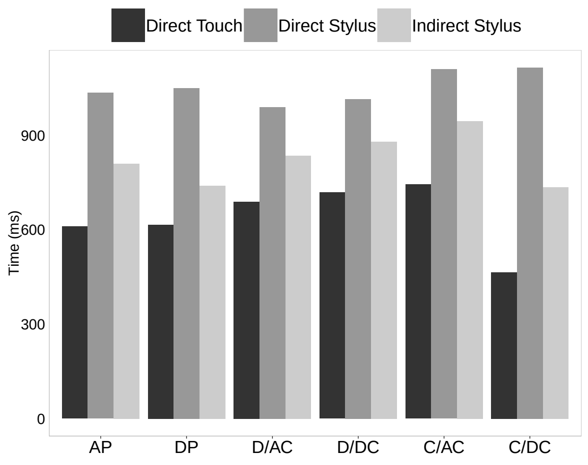
We conduct a post-experiment questionnaire to collect participants' subjective opinions on the easiness and accuracy of touch crossing vs tapping, directional targets vs amplitude targets, discrete crossing vs continuous crossing. As a result, 8 out of 12 participants said tapping was easier, however, interestingly, 8 out of 12 said crossing was more accurate. 10 out of 12 participants said targets with a directional constraint were easier and more accurate. 7 out of 12 participants felt discrete crossing was in general more accurate than continuous crossing, but both were among the same degree of difficulty.

2.2.3 Discussion

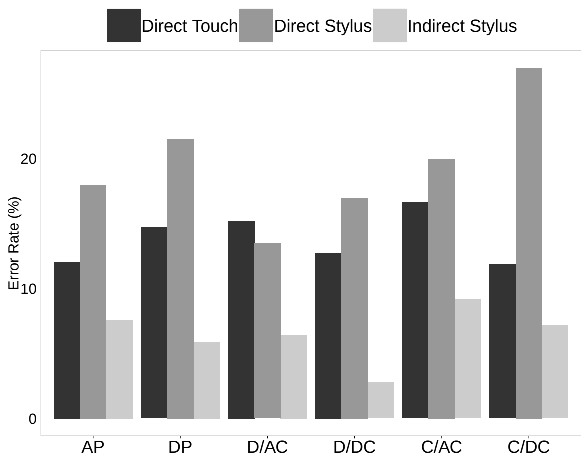
The subjective results from our questionnaire align with our main quantitative finding: in terms of time and error rate, touch crossing is as good, or better, than pointing with directionally constrained targets; in Accot and Zhai, and Dixon et al. stylus crossing studies [16, 3], orthogonal targets also have the highest performing; considering only orthogonal targets, continuous crossing is 33% faster than pointing in our experiment.

Comparison of Touch Crossing to Stylus Crossing

We present a meta comparison in this section among our results of touch crossing, Accot and Zhai's indirect stylus crossing, and Forlines and Balakrishnan's direct stylus crossing, in order to contextualize touch crossing performance. This is possible because first, our experiment and Forlines and Balakrishnan's experiment are near replications of Accot and Zhai's original study and second, we pick up measurements that overlap most IDs and have similar range of previous work. Note that our comparison uses quantitative results determined by analyzing figures in both papers, so quantities are approximate. In addition, recall that we reproduce 6 out of 8 task IDs used in Accot and Zhai and 4 out of 8 IDs from Forlines and Balakrishnan. Also we remove higher ID tasks to accommodate known limitations of touch input, but we also inserted an additional medium high ID as described in Section 2.2.1. To sum up, our intention here is to contextualize rather than to make definitive conclusions.



(a) Time comparison



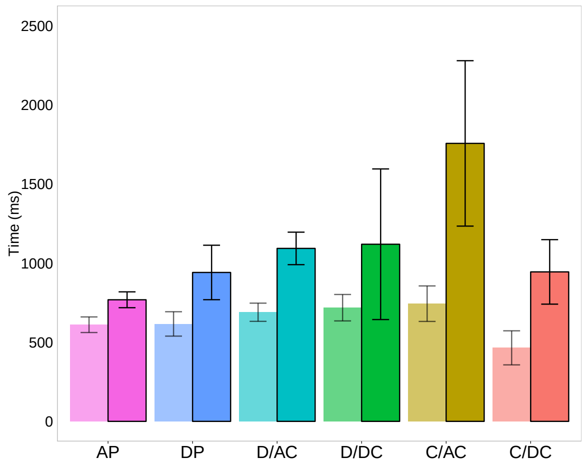
(b) Error comparison

Figure 2.10: Comparison with previous work: our experiment 1 ***direct touch***; Forlines and Balakrishnan ***direct stylus*** [19]; Accot and Zhai ***indirect stylus*** [3]

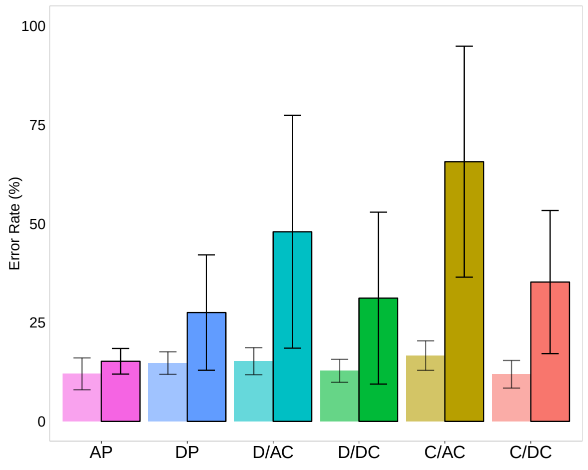
As shown in Figure 2.10a, task time for touch crossing compares favorably with both indirect stylus crossing and direct stylus crossing. While indirect stylus and touch have a similar performance trend across tasks (with *C/DC* performing fastest), the trend for direct stylus favors crossing with amplitude targets. Error rates (Figure 2.10b) are lower for indirect stylus in all cases, but touch crossing error rates are lower than all direct stylus crossing tasks except *D/AC*. In short, the trend is encouraging touch crossing and tapping appear faster.

Cursor Feedback Visualization

Because touch cursor feedback is rarely used in commercial devices, we originally ran a version of Experiment 1 without any cursor feedback. After 6 participants (3 female) using exact the same configuration of the main experiment, we found error rates were extremely high. Participants were frustrated and one of them even "gave up". Thus, we abandoned this approach and considered this as a "no feedback pilot" for Experiment 1, which enables us to compare the time and error rates between touch crossing with no feedback and with visual feedback. In our experiment 1, we use a circular cursor visualization, which decreases time and decreases errors as shown in Figure 2.11, but it remains unclear if other types of feedback are also effective. In our second main experiment which we are about to examine, we test four types of cursor feedbacks on crossing performance. Details can be found in next section (Section 2.3.3).



(a) Time comparison



(b) Error comparison

Figure 2.11: Comparison of time by *TASK* type: experiment 1 data ***with feedback*** (light colors); data from a pilot evaluation ***without feedback*** (dark colors)

2.3 Experiment 2: Touch Crossing Modelling

2.3.1 Motivation: Poor Fitts' Law Modeling for *D/DC*

To further validate the fundamental performance of touch crossing, we model all *TASK*s in Experiment 1 with Fitts' Law (see 2.1.1). As shown in Table 2.4, five out of six tasks are well modeled with Fitts' Law with high fitness values. However, the model for *D/DC* achieves only $r^2 = 78\%$, which is unexpected. In Accot and Zhai's pen crossing experiment, all six tasks are all well modelled with Fitts' Law as shown in table 2.3.

Task	AP	DP	D/AC	D/DC	C/AC	C/DC
Indirect Pen	99.8%	98.6%	99.4%	97.5%	99.5%	98.4%
Touch Crossing	96.5%	91.9%	93.5%	77.8%	94.5%	95.8%

Table 2.3: Fitts' law models for Accot and Zhai's indirect pen crossing and our touch crossing. Note that the fitness value of *D/DC* (highlighted) is low.

Task *D/DC* is really important in practice because discrete crossing is essential in real-life applications. Although continuous crossing is shown to be a promising selection approach according to our first experiment, users are likely to make discrete selections on distinct visual targets that can be manipulated independently of each other, for example, selecting multiple files that are not listed adjacently. Additionally, majority of the participants from Experiment 1 commented that discrete crossing preformed more accurate than continuous crossing. Furthermore, discrete crossing action must be performed at the start and the end of all crossing interactions, even when crossing multiple goals in a continuous action. In conclusion, the utility of both selection approaches are important, and the poor modeling with Fitts' law is a concern. It would be great if we can find a more accurate predictive model for task condition *D/DC*.

$$
\begin{gathered}
\mathit{AP}: T = 24.45 + 158.12 \times \mathit{ID}, r^2 = 96.48\%, p < 10e^{-6} \\
\mathit{DP}: T = 75.29 + 147.16 \times \mathit{ID}, r^2 = 91.87\%, p < 10e^{-4} \\
\mathit{D/AC}: T = 141.03 + 147.62 \times \mathit{ID}, r^2 = 93.53\%, p < 10e^{-5} \\
\mathit{D/DC}: T = 247.81 + 126.89 \times \mathit{ID}, r^2 = 77.79\%, p < 0.001 \\
\mathit{C/AC}: T = -51.98 + 211.39 \times \mathit{ID}, r^2 = 94.54\%, p < 10e^{-5} \\
\mathit{C/DC}: T = -195.39 + 173.92 \times \mathit{ID}, r^2 = 95.81\%, p < 10e^{-6}
\end{gathered}
$$

Table 2.4: Fitts' law models for all tasks

Inspired by FFitts Law [13], a stronger model for touch pointing interactions with small targets (see details at Section 2.3.2), we design the second experiment and test the predictive power of FFitts Law on touch crossing interactions. Our primary goal is to apply Bi et al's FFitts law, a modification of Fitts' law [1] to directionally constrained crossing to attain a better model for *D/DC*. Our motivation is that FFitts law explicitly considers precision with touch input, and this is a possible reason for the poor fit of D/DC with Fitts' law. In addition, since cursor feedback affected performance in Experiment 1 (see discussion in Section 2.3.3), we also examine the effect of different types of feedback in more detail in our second experiment.

2.3.2 Fitts' law and FFitts law

Fitts' law models the trade-off between speed and accuracy in human performance using a single distribution of selection endpoints. This distribution is a function of the nominal target distance D and the effective target width W_e, where $W_e = \sqrt{2\pi e \sigma}$ and σ is the empirically observed variability of selection endpoints. However, when using an input modality like touch, a portion of endpoint variability is due to the absolute precision of the finger. Specifically, observations, especially those with larger variability, indicate that a portion of the variability in endpoints is independent of the performer's desire to follow the specified precision and cannot be controlled by a speed-accuracy trade-off. This portion of variability reflects the absolute precision of the finger input. In other words, the observed variability in the endpoints may originate from two sources: a) the relative precision governed by the speed-accuracy trade-off of human motor systems, which can be modeled by Fitts' law, and b) the absolute precision uncertainty of the finger per se.

FFitts law utilizes this characteristic of finger input, and breaks σ into two endpoint distributions: one resulting from the relative speed-accuracy trade-off σ_R and the other resulting from absolute accuracy σ_A. Assuming central distributions, these are related as $\sigma^2 = \sigma_R^2 + \sigma_A^2$. Thus, index of difficulty from FFitts Law is expressed as Equation 2.3.

$$ID_f = \log(\frac{D}{\sqrt{2\pi e(\sigma^2 - \sigma_A^2)}} + 1) \tag{2.3}$$

To apply FFitts law, we need two distributions σ_R and σ_A. From our Experiment 1, we are able to formulate empirically measured distributions of endpoints σ. Now we need to set up another experiment to obtain σ_A. Our primary goal is to test the applicability of FFitts law on touch crossing performance. Since all tasks can be well modeled by Fitts' law except *D/DC*, and given that D/DC is a directionally constrained task, we design experiment 2 to measure σ_A for directionally constrained tasks. Thus, the σ_A calculated from Experiment 2 would only make sense for *C/DC* and *D/DC*. Note that Bi et al. use a similar controlled experiment design to find σ_A for touch tapping.

2.3.3 Cursor Feedback Type

To measure the influence of visual feedback on crossing performance, we also test four types of cursor *FEEDBACK* (Figure 2.12) in our Experiment 2. *FEEDBACK C* is a gray dot at the touch position with a radius 4.6mm transparent circle outlined in a 1 px blue stroke. This is the same cursor visualization used in Experiment 1. *FEEDBACK T* is a 1px thick blue line representing the trail of 35 recent touch points. *FEEDBACK CT* combined the circle cursor and trail stroke. *FEEDBACK N* has no feedback.

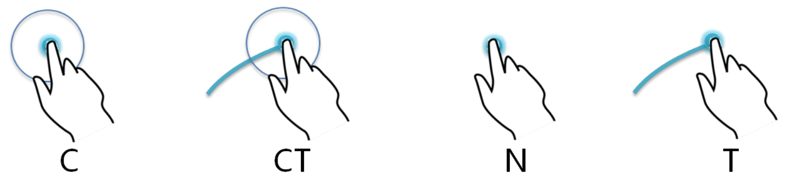


Figure 2.12: Cursor *FEEDBACK* type: *C* for cursor only; *CT* for cursor and trail; *N* for none; *T* for trail only.

2.3.4 Experiment Design

Task and Stimuli

We tailor the "Finger Calibration Task", used in Bi et al.'s first experiment, to directionally constrained crossing. A single 0.6mm crossing target is shown at the center of the display with no distance amplitude specified. The target size follows Bi et al.s recommendations: it is very small, but still legible (2px on our display) and it requires the highest possible precision to achieve. Two vertical bars are placed 10px to either side of the small target to remind participants to cross horizontally, and they are asked to cross from left to right to be consistent with the directionally constrained tasks in Experiment 1.

In each trial, the participant first tap a larger start button to force a break, and then cross the small target. We ask participants not to rush and focus on accuracy: measuring σ_A is not about speed [13]. A magnified view of vertical deviation from the target are shown at the top of display and an error sound plays if target is missed. Both are intended to help participants focus on accuracy. Errors are determined in the same way as Experiment 1.

Participants and Apparatus

Twelve right-handed participants were recruited (6 female), ranging in age from 19 to 29. All reported experience with touch screen devices. The same apparatus as Experiment 1 was used.

Procedure and Design

The experiment design is within-subjects, repeated measures, and full factorial. For each of the four cursor *FEEDBACK*s presented in randomized order, participants are instructed to perform 10 practice trials and then 10 measured trials. Participants are also asked to finish a post-experiment questionnaire and rank the four *FEEDBACK* types in terms of ease-of use and accuracy. The experiment took 10 minutes of one participant on average.

2.3.5 Results and Discussion

Data Processing

4.4% (20 out of 453) data points more than three standard deviations from the mean were removed as outliers. Note that vertical crossing and trials crossing from right to left were removed before testing outliers (27 out of 480), as they violated our design assumptions.

Absolute Precision Distribution (σ_A) and *FEEDBACK*

The distribution of deviations from target center in Experiment 2, called σ_A, is the standard deviation of all intersection points between the touch path and the vertical line through the target. Smaller σ_A values reflect higher absolute accuracy as we discussed before.

FEEDBACK	σ_A(mm)	Errors(%)	Speed (m/s)
C	1.12	38.3	0.22
CT	0.96	32.5	0.16
N	1.59	85.8	0.23
T	1.03	35.0	0.16

Table 2.5: σ_A, error rate, speed by *FEEDBACK* type

There is a main effect of *FEEDBACK* on σ_A ($F_{3,33} = 3.416, p < 0.003$). However post-hoc analysis finds no significant difference in means (Table 2.5). During the experiment, participants in the *N* condition appeared to cause appreciably more audible error tones than the rest. An analysis of error rate shows a main effect of *FEEDBACK* on error rate ($F_{3,33} = 16.53, p < 0.005$). Post-hoc analysis finds the error rate of N (85.8%) is significantly different and is higher than all other feedback types, while no difference has been found between *C*, *CT* and *T*. In the post-experiment questionnaire, ten out of twelve participants voted *N* as most difficult and least accurate.

The selection deviation distributions for each feedback type provide more insight into reasons why *N* has such a high error rate, but more reasonable σ_A (Figure 2.13). *C*, *CT*, and *T* have modes near the target center, so distributions are in the target. The mean of *N* is below the target, so the distribution σ_A does not fall within the target. This consistent offset is similar to Holz and Baudisch's observation of perceived versus actual input positions with touch tapping [28].

Participants were instricted to do the whole experiment as accurate as possible without any limitations on speed. Table 2.5 includes the instantaneous velocity of the moment participants are trying to cross the target. All speeds are really close without any significantly differences in regards of *FEEDBACK*. We can further confirm that the accuracy performance of these four cursors are only dominated by the *FEEDBACK* types.

2.3.6 FFitts Law for Directionally Constrained Crossing

We apply FFitts Law to the data from the two directionally constrained tasks in Experiment 1, *D/DC* and *C/DC*, using Bi et al.'s procedure [13]. Our intent is to improve the *D/DC* model, but we include *C/DC* for completeness. We use the σ_A value of feedback *C* since it was used in Experiment 1. The real distributions of endpoints σ can be obtained from Experiment 1. We calculate σ values using the same method as σ_A, but on the actual target widths of Experiment 1.

At first, modeling with FFitts Law was very poor for *D/DC* with $r^2 = 58.53\%$, but reasonable for *C/DC* with $r^2 = 84.31\%$, a little bit lower than Fitts' Law. On closer

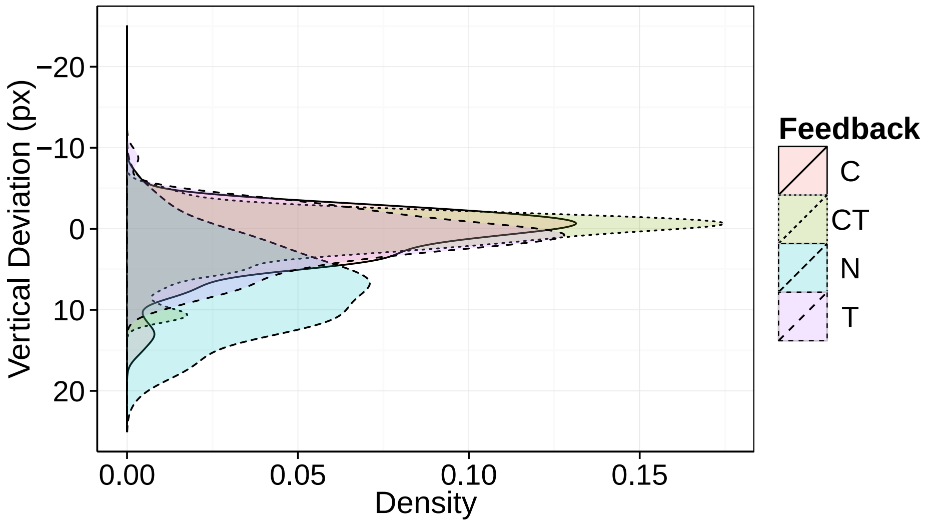


Figure 2.13: Distribution of crossing points, by *FEEDBACK*

inspection of the *D/DC* data (Table 2.6), we noticed that the highest error rate (30.6%) and longest time was for ID = 6, the combination of W = 3.2 mm and D = 203.3 mm. Looking at the distribution of crossing points for *W* = 3.2 mm revealed a slight bimodal distribution (Figure 2.14), which was not seen with *W* = 3.2 mm and the short distance *D* = 50.8 mm. Note that FFitts law assumes a unimodal distribution of selection points, so it is not surprising that FFitts law did not work well with this point.

To justify our hypothesis, we remove the unusual *ID* = 6 points and re-run the regression with FFitts law with the remaining nine *ID*s. This time the r^2 for *D/DC* improves from 75.26% with standard Fitts' law to 84.3% with FFitts law (Figure 2.15a). The fitness for *C/DC* remains similar at $r^2 = 85.9\%$ (Figure 2.15b). Admittedly, a higher r^2 value is desirable, but our exercise provides evidence for the applicability of FFitts law to crossing.

2.4 Summary

In this chapter we present two experiments to validate the fundamental performance for crossing-based selection tasks with direct touch input. Experiment 1 is a close adaptation of Accot and Zhais indirect stylus crossing experiment. We test six different types of conditions and conclude touch crossing follows similar trends of indirect stylus crossing: touch crossing task time is faster or equivalent to touch pointing; continuous selection of large orthogonal crossing targets is most effective; and continuous selection of small collinear targets is least effective.

D (mm)	*W* (mm)	ID (bits)	Error Rate (%)
50.8	3.2	4.1	19.6
50.8	5	3.5	14.1
50.8	6.4	3.2	7.6
50.8	12.7	2.3	1.8
50.8	25.4	1.6	6.7
203.3	**3.2**	**6.0**	**30.6**
203.3	5	5.4	14.2
203.3	6.4	5.0	10.8
203.3	12.7	4.1	2.0
203.3	25.4	3.2	0.4

Table 2.6: Error rates of *D/DC* by D-W combination

We notice that, unlike indirect stylus and mouse crossing, not every kind of direct touch pointing performance is modeled accurately with standard Fitts law. Thus in our Experiment 2, we apply "FFitts" law, used previously for touch pointing with small targets, in order to obtain better performance fitness on discrete touch crossing with a directionally constrained target.

In addition, in Experiment 1 we notice that visual feedback has an impact on touch crossing performance, this has been further confirmed in Experiment 2, where visual touch feedback is shown to have a strong effect on absolute accuracy of finger input. Our work serves to validate crossing performance for touch and demonstrates a new application of FFitts law. This new empirical evidence provides necessary support for crossing-based interaction techniques in touch interfaces.

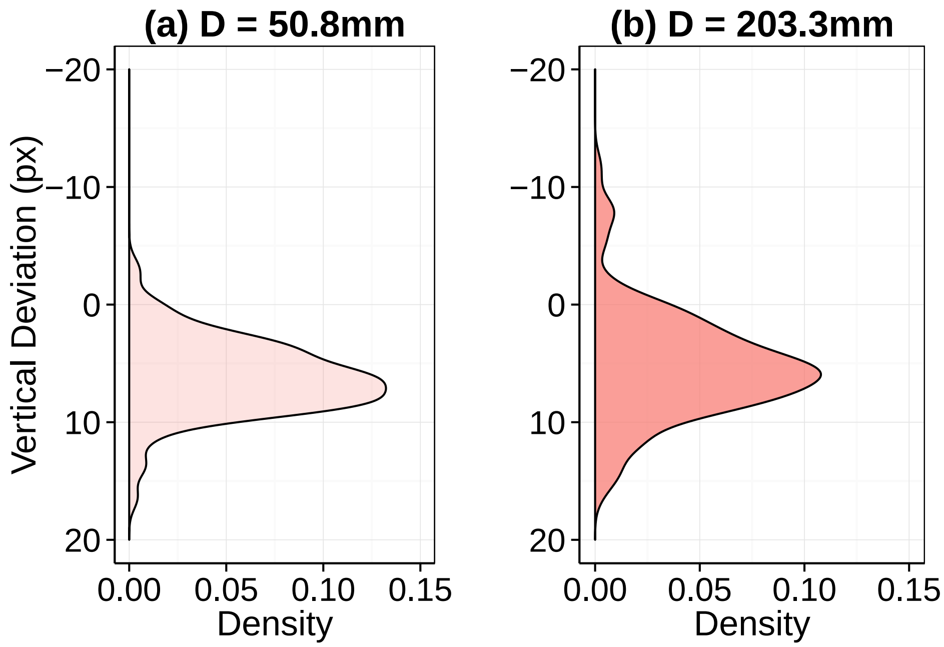
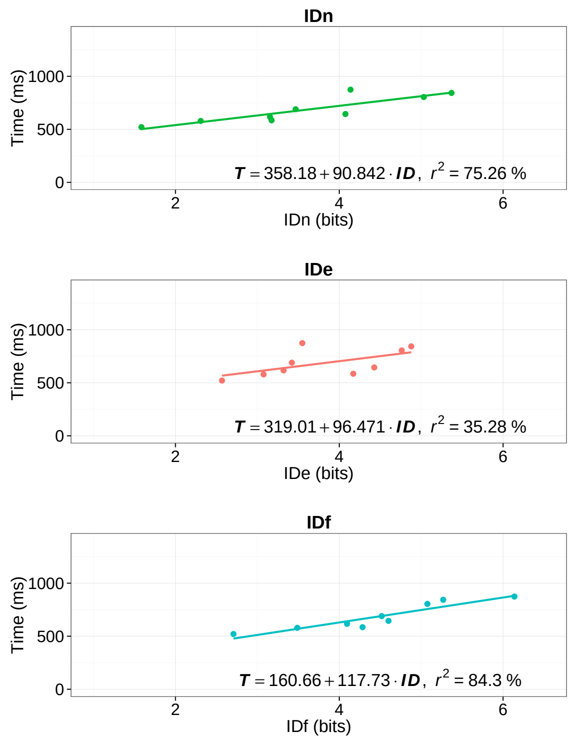
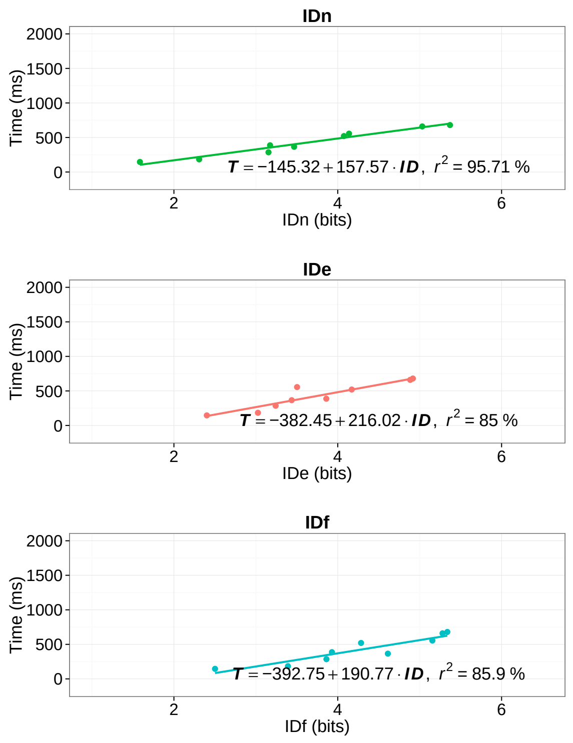


Figure 2.14: Crossing point distributions for *D/DC* with W = 3.2mm for the two distances



(a) *D/DC*



(b) *C/DC*

Figure 2.15: Modeling *D/DC* and *C/DC* using: (ID_n) Fitts' law; (ID_e) Fitts' law with effective width; (ID_f) FFitts law

Chapter 3

Systematic Exploration of Pin-and-Cross

ADDITIONAL INFORMATION
This chapter is accompanied by a video which demonstrates the experiment setup of the formative study described in this chapter.
The video is available at: `http://youtu.be/NK-yRta38RA`

With touch crossing performance validated in the previous section, now we are able to further explore other novel forms of crossing interfaces. We propose *pin-and-cross*, where static touches ("pins") are combined with nearby crossing selection (selecting a line target by stroking through it). For example, with a pin-and-cross context menu, one finger pins a graphical object while another finger on the same hand crossed a line target to select a menu item (Figure 3.1). Targets can be crossed in either direction for different items and in the meantime, the object can still be dragged with the pin finger without entering into special modes.

Obviously pin-and-cross has some overlap with touch crossing actions. We also notice that pin-and-cross is not using exactly the same muscle groups as touch crossing due to the additional effort of performing "pinning". Evidence produced from the previous study inspire our exploration of pin-and-cross. Most importantly, it focused on the overall performing characteristics of touch crossing and serves for more generalized design problems.

This chapter studies the characteristics of human performance with unimanual pin-and-cross techniques under the context of command invocation. First we identify the research scope for pin-and-cross techniques, and explore previous approaches to multi-touch command invocations with one-handed or two-handed input. Next we conduct a controlled experiment with four unimanual pin-and-cross techniques using one or two pin touches, after either a tap or a drag, with a set of radial targets. Lastly, we analyze the input kinematics and performance matrix generated from the study, and produce guidelines for target length and angle based on time, error, and subjective preference, for pin-and-cross menu design.

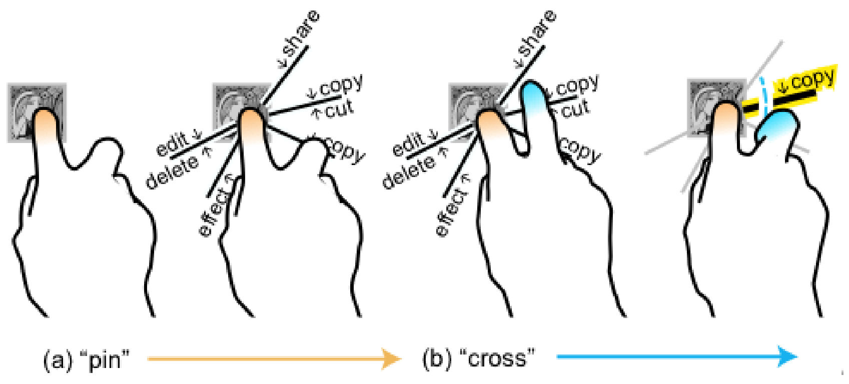


Figure 3.1: Unimanual *pin-and-cross* context menu: (a) one finger pins an object; (b) another finger crosses a radial target.

3.1 Related Work

3.1.1 Touch Overloading

Graphical buttons are conventionally used to access different modes or commands, however buttons can take up valuable screen space, shift focus away from the object of interest, and introduce a round-trip movement cost [9]. Touch overloading [25] is proposed to remedy those problems and achieve space-conserving mode selection or command invocation. The idea is to discriminate between different types of touches to enable different functions. This can be achieved using bimanual input, for example, adding a simultaneous touch with the other hand. However, bimanual input requires sufficient space and the usage could be greatly limited on devices with small touch area such as smart phones or tablets. Unimanual touch overloading, on the other hand, is more adaptable and better able to cope with different scenarios. Examples of commonly used unimanual overloading include dwell time (e.g. tap to select, press-and-hold to open menu), gestures (e.g. tap to select, swipe left or right to delete), and multiple fingers (e.g. one finger tap to select, two fingers to pan and zoom). Alternatives, touch can be overloaded with additional hardware sensors such as pressure [45] ,shear [24], sound [25] and finger identification [7], or with dedicated activation widgets [23], or by leveraging non-conflicting input spaces such as fast but distant taps [26] and bezel swipes [29]. Note that all approaches are mutually compatible and can be combined to further increase the input space for touches. Pin-and-cross is built on top of this idea, where touch input space is expanded by combining static touches with dynamic crossing selections.

3.1.2 One Handed Command Invocation with Discrete Touches

Different types of touches are most commonly used as approaches to achieve efficient command invocation. Wu and Balakrishnan's RoomPlanner [49] demonstrates usage of unimanual two-finger techniques on tabletop. For example, they use thumb to position a toolglass then select items by index finger. They also implement freeform object rotation by specifying center with thumb and controlling angles with index finger. TapSense [25] classifies different types of touches, such as fingernail, knuckle, or fingertip, by observing sound frequency spectrum.

Banovic et al. designs a partial Pie Menu [7] that can be operated with two fingers interactions including two taps on same finger, pin and tap (simultaneous), and pin then tap (sequential). Sequential and simultaneous selections on average perform faster than two taps, however lead to higher error rates. A pilot study is performed to determine target measurements with limited variables being examined (only angle and radius). Note that like marking menus, touches in Banovic et al. partial pie menu must be disambiguated from other direct manipulations like dragging, selecting two objects, or beginning a two-finger transformation. One commonly used solution is to "press-and-hold" an object to enter menu selection mode and trigger contextual commands (e.g. sharing, rotating, or deleting). The "hold" is able to disambiguate but brings extra time penalty. "pin and cross" identifies the object of interest with one finger pinning on the display similar to Banovic et al., however pin-and-cross uses a crossing stroke performed by a nearby finger to disambiguates itself from other manipulations. Additionally, certain fingers (index and thumb) are forced to function as the pin finger in Banovic et al. design on table tops, while pin-and-cross targets on tablet apps and has no restrictions on the usage of fingers.

Techniques sharing similar ideas of Banovic et al.'s design includes distant tap gestures such as Ta-tap/Ta-ta-tap [26] and Bezel-Tap [44]. Slide Rule [31] uses a second finger tap to select the current menu item in an audio. FastTap [23] utilizes one-handed sequential selection to achieve fast command invocation.

3.1.3 One Handed Command Invocation with Continuous Strokes

Target selection utilizing strokes instead of discrete taps further enriches the input space of command invocation. Marking menus [36] [35] allow users to make selection on a radial menu by drawing a mark toward the target. Variations of Marking Menu include zone/polygon menus [50], compound marking menus [53], and multi-touch marking menus [38]. In the multi-touch marking menu proposed by Lepinski et. al's, chorded directional gestures are used for drawing marks. Specifically, multiple fingers contact the screen and hand swipes in a direction to activate a command. Fingers are identified by comparing touch locations relative to a bounding box of user's hand, captured by tracking cameras.

Crossing-based selection, where users draw a stroke through a boundary target to invoke selection, has been explored as an alternative strategy for menu designs. Luo and Vogel [39] demonstrate the applicability of crossing by comparing tapping with crossing, and examining various crossing conditions. Motivated by potential benefits of crossing selections, AttachedShock [55] creates an expanding wave pattern on moving targets under mobile navigation system, where users cross the wave to achieve higher accuracy. X-O Arch Menu [47] combines the design of hierarchical pie menu and crossing selection. Menu items on deep hierarchy can be selected by swiping back-and-forth without lifting fingers, similar to target reverse crossing proposed by Feng et. [18] with the mouse. In the study of AttachedShock and X-O Arch Menu, crossing-style selection is reported as an alternative to improve time and precision performance of command invocation.

3.1.4 Two Handed Command Invocation

Some of the previous approaches to multi-touch command invocation require two hands. Bailly et al. [6] combine static touches with dynamic dragging, and demonstrate a set of two-handed multi-finger gestures using Finger-Count interactions, where menus are triggered with touches on one hand, fingers on the other hand drag up and down to control parameters such as locations of the menu item. BiTap and BiPad [49] perform bimanual taps, gestures and chords by holding a tablet. Kin et al. [33] introduce a two-handed simultaneous marking menu on mobile devices, where marking stroke are drawn by both hands together. HandyWidgets [54] treats the segment space between two fingers as a crossing target, and invokes command by crossing the target with the other hand. As we have discussed before, two-handed input would become challenging or impossible on devices with smaller displays or under a mobile context, while one-handed command invocation is more universal.

Various exciting work has been proposed using discrete touches, strokes to achieve efficient command invocation with one hand or two, but to the best of our knowledge, unimanual interactions combing touches and strokes is still under-explored. In the following section we contribute a systematical study to understand the human capability on performing multi-touch and stroke simultaneously with one hand. Results are generalized to a set of design guidelines for future exploration on pin-and-cross based interfaces.

3.2 Formative Study

3.2.1 Experiment Design

The primary goal of this experiment is to explore fundamental performances of pin-and-cross and its variations in order to develop design guidelines and recognition algorithms, which we will reveal at the conclusion of this chapter and in Chapter 5. The study measures input kinematics, performance metrics, and subjective preferences for different radial crossing targets and pin-and-cross technique variations. We control for "pinning" with one or two fingers and whether a pin occurs immediately after a tap or after a drag. Note that we use "digit" to represent any of the five fingers of hand including the thumb. Results of this study serve to: validate different pin-and-cross technique variations including desired pin digits (3.2.2); produce guidelines for crossing target length and angle based on time, error, and subjective preference (3.3); and provide empirical thresholds for recognition algorithms (5).

Participants

Twelve right-handed participants were recruited (3 female), ranging in age from 22 to 37 years old. All reported experience with touch screen devices.

Apparatus

The experiment was performed on a Google Nexus 10 multi-touch tablet running Android 4.4. The $264 \times 178mm$ display has a resolution of $2560 \times 1600px$, a density of $11.8px/mm$ (300 PPI). The software was written in Java using the Android SDK. Participants were seated and used the landscape-oriented tablet laid flat on a table. A video camera recorded the hand and tablet from above and all input events were logged to a file.

Tasks and Stimuli

The basic task is to touch and hold a pin target with one or two fingers and simultaneously use another available finger on the same hand to cross through a line target (50mm

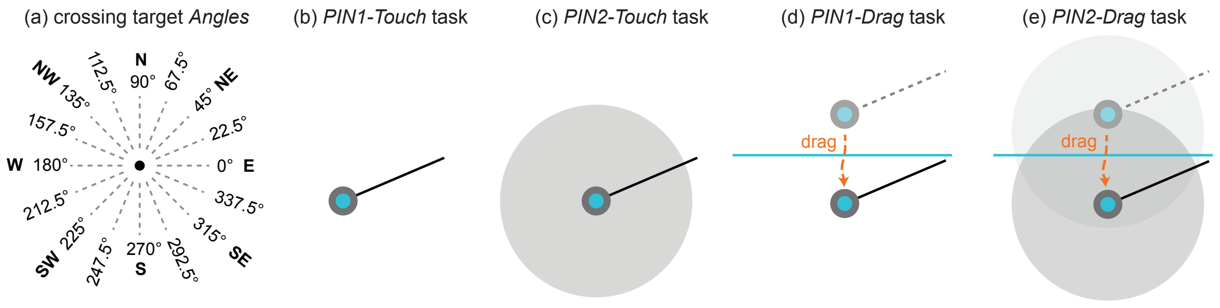


Figure 3.2: Experiment task: (a) 16 crossing line target locations in a radial array 22.5° apart; (b) *1PIN-Touch* task requires a one finger touch on circular target for the pin, then cross a single target with another finger (22.5° target shown for all tasks); (c) *2PIN-Touch* task requires a one finger touch on circular target and a second finger touch in the gray area for a two finger pin before crossing; (d) *1PIN-Drag* requires one finger contact on circle, then a downward drag below a line before crossing; (e) *2PIN-Drag* requiring same two finger pin to perform downward drag below line before crossing.

in length, 1px thick) arrayed around the pinning target. The crossing target is always visible and anchored to the pin target. We use the radial line target to guarantee that selection is initiated with a direction constrained crossing. Performing pin-and-cross with an amplitude constrained crossing is similar to a zooming action, which is against our original intention to disambiguate pin-and-cross with existing unimanual multi-touch techniques. Additionally, combining static touch with the amplitude constrained crossing may be awkward to perform and raise the issue of physiological discomfort.

We control for 3 independent variables to manipulate this basic task.

- *Angle*. We pick up 16 equally spaced radial locations as crossing targets (Figure 3.2a). These locations are intended to uniformly sample possible target angles, not necessarily function as exact angles in an implementation. We define *Angle* 0° to be straight right (East) and increasing counter-clockwise in 22.5° increments such that *Angle* = 0°, 22.5°, 45°, 67.5°, 90°, 337.5°.
- *Pin Number* refers to the number of fingers used to pin: one (*1PIN*) or two (*2PIN*).
- *Pin Type* specifies if the pin occurs after a *Touch* or after a *Drag*. The drag condition simulates situations where pin-and-cross are used together with actions like dragging objects, scrolling, or two-finger transformations.

The four combinations of *Pin Number* and *Pin Type* form four types of pin-and-cross tasks (Figure 3.2b-e):

- *1PIN-Touch*. Participants touch and hold a 33mm diameter blue and dark gray pin target, then stroke through the crossing target with another finger.

- *2PIN-Touch*. Similar to *1PIN-Touch* except two pin fingers are required, where one finger must touch the blue and dark gray target as before, while the second finger touches and holds anywhere in an enclosing 101mm light gray circle.
- *1PIN-Drag*. Pin target is centered 37mm from the top of the screen. The crossing target is shown, but cannot be crossed until the pin target is dragged with one finger below a horizontal solid line, which is placed 42mm from the top of the screen. Selection is the same as *1PIN-Touch*. Note that we only test one direction for *Drag* tasks. This is an acceptable limitation, given that our interest is what happens after dragging.
- *2PIN-Drag*. Similar to 1PIN-Drag except two pin fingers are required (similar to *2PIN-Touch*) when dragging object towards the horizontal line.

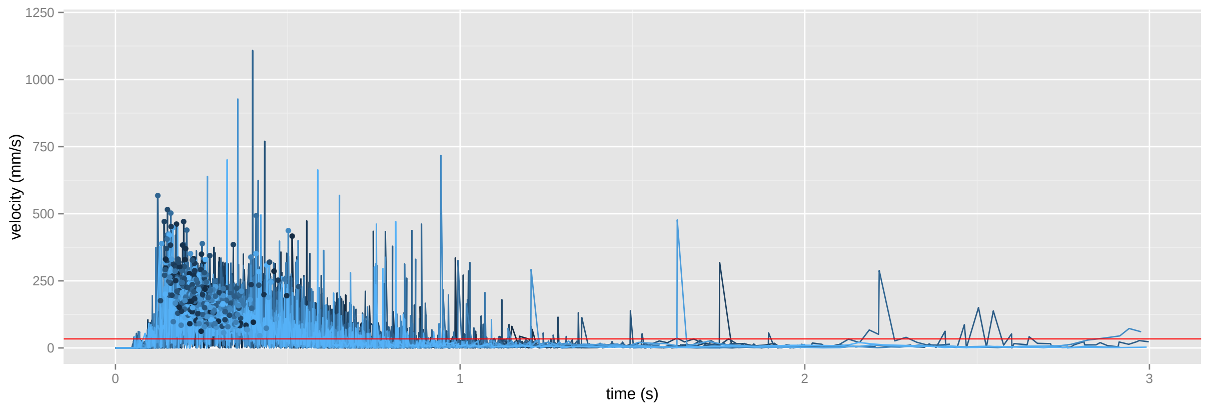
We intend to not control for other variations of the task to discover preferred pin-and-cross behavior and more natural kinematics. Participants can choose which finger(s) to use for pinning and for crossing. Only one crossing target is shown at a time so participants can choose how much landing and takeoff area they need to cross. The crossing target length is long enough to be unconstrained to permit participants to choose the preferred location for the cross [16]. For *Drag* tasks, participants naturally stopped the drag motion before starting the crossing selection. For *2PIN* tasks, the extended light grey circle is large enough to permit participants to choose the most comfortable location for their second pin finger. Displaying the crossing target from the beginning of the task allows time of planning on which fingers to pin to reach the target. Participants were instructed to perform all tasks as quickly and accurately as possible.

Task Errors and Time

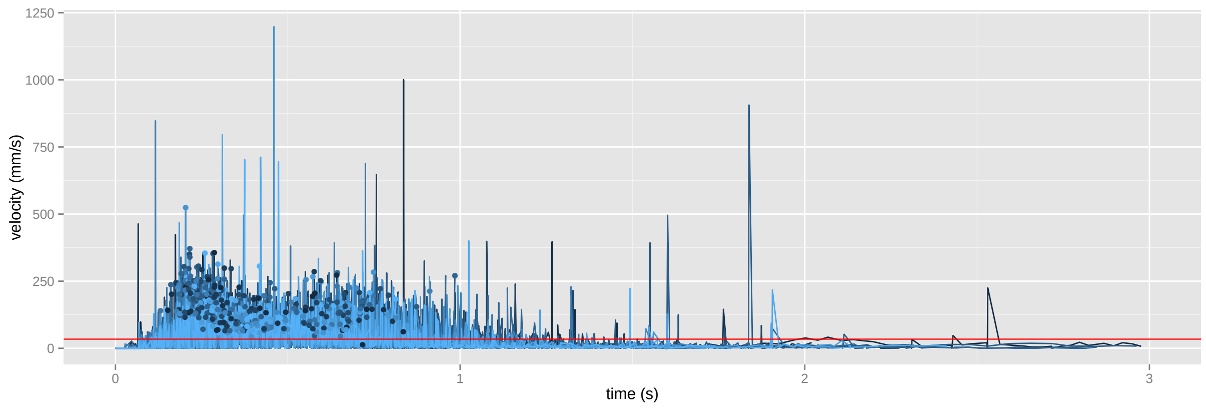
A *Touch* task was considered to be successful if the correct number of fingers (one or two) were touching the pin target(s), while a line segment within the crossing stroke intersected with the crossing target. All fingers would be released then without subsequent touches. *Drag* task was similar except that the correct number of fingers (one or two) should drag the pin target(s) below the given horizontal line, and remained touching while starting a crossing selection.

Note that two types of error, pinning errors and crossing errors, are possible given the criteria above. Our primary interests are crossing errors since pinning errors are more a result of the controlled task. When any error occurred, a tone sounded and the same trial was restarted. We recorded the number of errors, but only used the error-free trials in non-error rate analysis in the later sections.

Task time is defined as the duration between the first pin action and when the crossing stroke intersects the target. For *Touch* tasks, the first pin is unambiguous. For *Drag* tasks we are interested in pin-and-cross actions after a drag. To determine the moment



(a) Task *1PIN-Drag*



(b) Task *2PIN-Drag*

Figure 3.3: Pin finger velocities for *Drag* tasks. Threshold is labeled with horizontal line in red color. Blue lines represent velocities of all pin movement, and blue dots shows when dragging pass the horizontal line. Darkness indicates the target position (*Angle*), where lighter blue show targets between $0°$ to $180°$, while darker blue are targets between $202.5°$ to $337.5°$ The timeline (x-axis) represents the period from the first pin touch to the end of a task. Values on the timeline are all calculated relatively to the time of a first touch within each task.

that a drag action ends, we looked at our log data and define the first pin action for *Drag* tasks to be when all pin touch velocities drop below $34mm/s$. Note that the threshold we are looking for should meet the assumption that it occurs after the horizontal line is passed, and before the start of a crossing selection, of which the pin velocity usually decreases to almost 0. As shown in Figure 3.3 where the thresholds is labeled with a horizontal line in red color, the threshold we picked up satisfies the assumption.

Design and Protocol

The experiment design is within-subjects, repeated measures and full factorial. All trials for each type of pin-and-cross Task were presented together, with Task order counter-balanced using a Latin Square. For each Task, participants received a short instructional demo, then performed 3 blocks of measured trials (the first block is considered a practice block). Each block presented all 16 crossing target *Angle*s in a random order. Short breaks were enforced at the end of each block. In summary:

4 *TASK*s (*1PIN-Touch*, *2PIN-Touch*, *1PIN-Drag*, *2PIN-Drag*) $\times$
3 blocks $\times$
16 *Angle*s $\times$
12 participants
= 2,304 data points

3.2.2 Results and Discussion

All tests use within-subjects repeated-measures ANOVA with Holm correction for post hoc tests.

Outliers and Learning Effect

We removed outlying data points with times more than 3 standard deviations from the task mean. This removed 1.04%, 2.0%, 1.91%, and 1.56% (6, 12, 11, and 9 out of 576) for the four tasks. Although *Block* 1 is considered practice, we include it when testing if blocks 2 and 3 have a learning effect. Testing each *Task* separately, block had a significant effect on time for *2PIN-Touch*, *1PIN-Drag*, and *2PIN-Drag* (all $F_{2,22} > 4.1, p < 0.031$). Post-hoc tests only found block 1 significantly slower ($p = 0.038$) for task *2PIN-Drag*, suggesting minimal learning effects for blocks 2 and 3. In subsequent analysis, we drop block 1 and aggregate blocks 2 and 3.

Time and Error by Task

No main effects were found for Task on crossing error rate, but a main effect exists for Task on time ($F_{3,33} = 26.97, p < 10e^{-8}$). Post hoc tests show *2PIN-Touch* (597ms) is slowest, then *1PIN-Touch* (428ms), then *1PIN-Drag* (348ms) is fastest ($p < 10e^{-4}$). No significant differences between *2PIN-Drag* (412ms) and other tasks ($p = 0.2$). The mean time across all *Task*s is 446ms (STD 173). All performance times are reasonable, which suggests that two-finger pins, though with more complexity, are still compatible with pin-and-cross techniques.

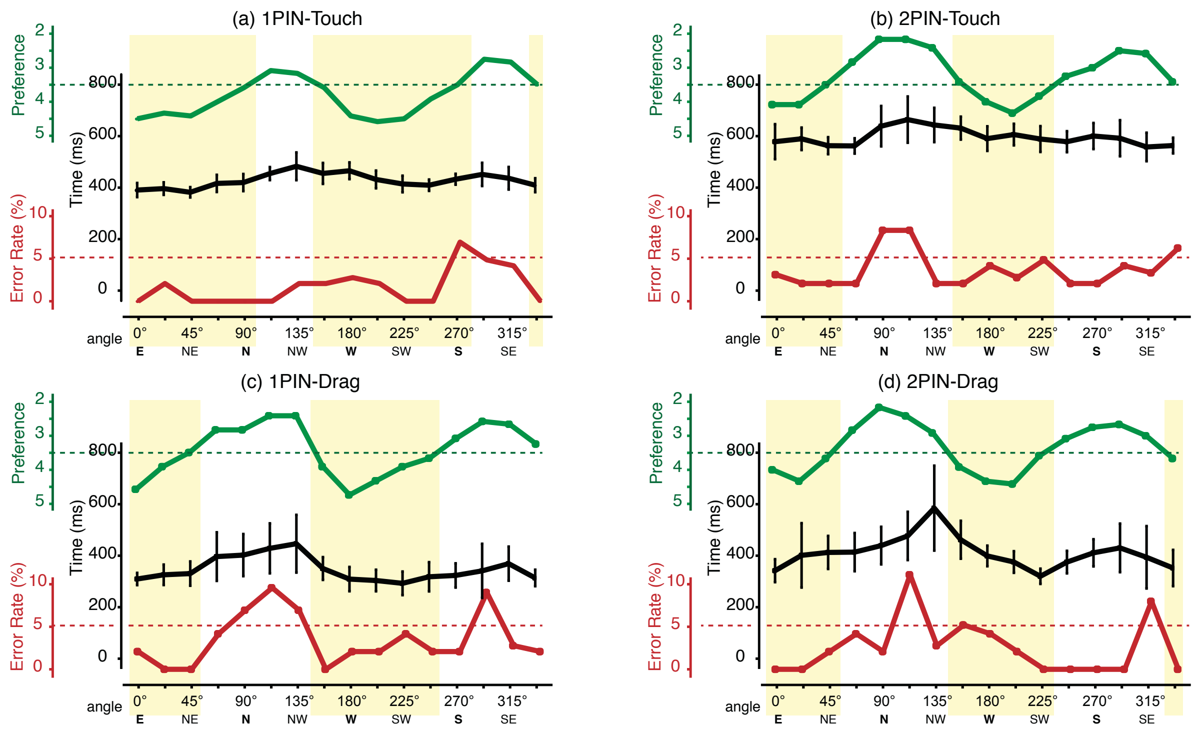


Figure 3.4: *Time*, *error*, and *preference* by crossing target *Angle* for each *Task* (lower values are better, 95% CI shown for time)

Time, Error and Preference by Angle

We examined task time, crossing errors, and subjective preference for 16 crossing *Angle*s for each *Task* independently (see Figure 3.4). These results motivate the design guidelines for pin-and-cross targets which we will reveal in section 3.3.

- *Time By Angle*. There is a main effect of *Angle* on time for *1PIN-Touch* ($F_{15,165} = 3.453, p < 0.0001$). Post-hoc test shows significant differences between the two horizontal crossing targets, where $0°$ (E) is faster than $180°$ (W). Additionally, $90°$ (N) and $45°$ (NE) are faster than $135°$ (NW). No effect has been found for *Angle* on time for *2PIN-Touch* ($F_{15,165} = 1.643, p < 0.068$). For *Drag* tasks, there is a main effect of time on *Angle* for both *1PIN-Drag* ($F_{15,165} = 2.403, p < 0.036$) and *2PIN-Drag* ($F_{15,165} = 2.679, p < 0.002$), however post-hoc analysis does not find any differences between individual angles. In the study we set up 16 targets and the lack of significant differences for *Angle*s may be partly due to the high number of pairwise comparisons, but Figure 3.4 suggests some increasing trend as targets approach $135°$ (NW).

- *Error By Angle*. *Angle* has a main effect on crossing errors for *2PIN-Drag* only

($F_{15,165} = 1.877, p < 0.029$), however without any pairwise differences. Similar to the effect of angle on time, this may be also due to high number of pairwise angle comparisons. By examining Figure 3.4, a general trend of increasing error is suggested when *Angle* approaches $90°$ (N) and $270°$ (S).

- *Preference by Angle*. Participants were instructed to rate their preference for each target angle on a numeric scale from 1 to 5 (5 is highly preferred, 3 is neutral), after the completion of all trial blocks for each *Task*. There is a significant main effect of *Angle* on preference for all *Task*s (all $F_{15,165} > 4.586, p < 0.001$). Post-hoc analysis reveals that, for *2PIN-Touch*, preferences are significantly higher for $22.5°$ (S-SE) compared to $112.5°$ (N-NW) (all $p < 0.05$); while for *1PIN-Drag*, $180°$ (W) is preferable compared with $112.5°$ (N-NW) (all $p < 0.02$). Figure 3.4 suggests a consistent pattern that angles near NW and SE are least preferred and angles near W to SW, and E to NE (close to the horizontal) are more preferred, which is in line with participants' comments that vertical targets are the hardest to reach.

Agreements on Pin Touch Digit(s)

Understanding which pinning digits are used naturally by human is valuable since it helps justify pin-and-cross's compatibility with existing direct manipulation tasks (e.g. index finger is commonly used to drag object). We use "agreement score" to determine the degree of consensus among participants on which pin digit(s) they used. We followed the same method proposed by Wobbrock et al. [52] as shown in Equation 3.1.

$$A_t = \sum_{D_i \subseteq D_t} \left(\frac{|D_i|}{|D_t|} \right)^2 \tag{3.1}$$

In Equation 3.1, A_t is the agreement score given a condition t, D_t is the set of all types of digits used by participants under condition t, D_i is a subset of identical digit(s) from D_t. For example, when examine the pin digits of a block of *1PIN-Touch* trials under condition "cross $0°$ target", 6 trials use index finger to pin, 5 trials use middle finger, 1 trial uses thumb, we compute the agreement score as follow:

$$A_{\text{cross } 0° \text{ target}} = \left(\frac{6}{12} \right)^2 + \left(\frac{5}{12} \right)^2 + \left(\frac{2}{12} \right)^2 = 0.45 \tag{3.2}$$

Alternatively, if we have all 11 trials use index finger to pin, only 1 trial use thumb, the agreement score would be:

$$A_{\text{cross } 0° \text{ target}} = \left(\frac{11}{12} \right)^2 + \left(\frac{1}{12} \right)^2 = 0.85 \tag{3.3}$$

Note that participants on the former condition have lower consensus on which pin finger to use, which leads to a lower agreement score. In the experiment we used the overhead video to record which digit(s) were used to pin the object during each trial. Using this data, we were able to calculate agreement score for which digit(s) were used. Agreement scores were calculated by *Task* for each *Angle*.

For *1PIN* tasks participants most often used the index or middle finger to pin depending on crossing target direction. The middle finger was used for targets pointing left (agreements for angles $147.5°$ to $270°$ are all greater than 0.5 for *1PIN-Touch*, and greater than 0.43 for *1PIN-Drag*). The index finger was used for targets pointing right (agreements all greater than 0.5 for angles $292.5°$ to $45°$).

For *2PIN* tasks, no obvious pattern was found beyond using two contiguous digits such as thumb and index, index and middle, middle and ring (sum of all agreements for these digit pairs are above 0.49). The index or middle finger was most often the pin anchor for targets pointing right (with middle or ring crossing), while the middle finger was most often used for targets pointing left or down (with index finger crossing). This suggests that two touch pin-and-cross menus could anchor left pointing crossing targets on the left touch, and anchor right pointing targets on the right touch.

Input Kinematics Analysis

Quantitative kinematic characteristics are useful for designing the optimum layout of crossing targets (e.g. how long, how much space between), and for determining thresholds that can be used in rule-based heuristics for recognizing pin-and-cross actions in the wild.

Using touch event logs, we calculated the following kinematic statistics with standard deviation and 95% confidence interval (all measurements are illustrated in Figure 3.5, and values are provided in Table 3.1):

- *Pin Distance (PD)* is the cumulative distance covered by the pin finger while the crossing finger moves from touch down (P_{start} as shown in Figure 3.5) to the point of crossing (P_{cross}).
- *Pin Speed (PS)* is the mean speed of the pin finger while the crossing finger moves from P_{start} to P_{cross}.
- *Crossing Length (CL)* is the distance from P_{start} to the crossing target line.
- *Crossing Speed (CS)* is the mean speed of the crossing finger P_{start} to P_{cross}.
- *Crossing Start Speed (CSS)* is the speed of the crossing finger at the beginning of the stroke, just after P_{start}. This provides a lower bound on crossing speed.
- *Crossing Angle (CA)* is the angle of the crossing stroke relative to the pin position.

- *Crossing Distance (CD)* is the distance from pin center to P_{cross}, in the later section 3.3 we used this measurement to determine the optimum target length.
- *Crossing Directions (CDir)*. A target can be crossed in two directions, clockwise or counter-clockwise. We used this measurement to determine the optimum command position in section 3.3.

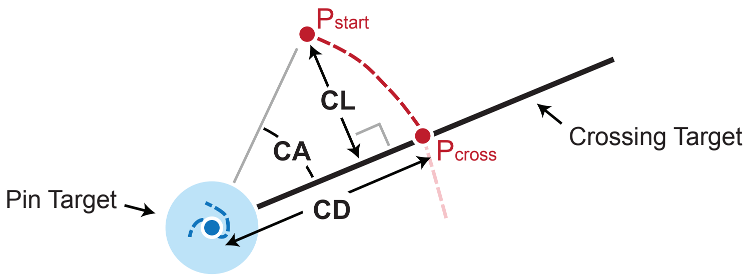


Figure 3.5: Kinematic measurements (crossing stroke is in red).

Measurement	*1PIN-Touch*	*2PIN-Touch*	*1PIN-Drag*	*2PIN-Drag*
PD mm	1.0(0.9) [0.9, 1.1]	1.1(0.9) [1.0, 1.2]	1.3(1.5) [1.2, 1.5]	1.5(1.7) [1.3, 1.7]
PS mm/s	15.1(18.0) [13.3, 16.9]	11.4(11.0) [10.3, 12.5]	17.9(19.6) [16.0, 19.8]	15.1(16.9) [13.4, 16.8]
CL mm	19.6(12.1) [17.4, 20.8]	18.6(11.7) [17.4, 19.8]	19.1(11.9) [17.9, 20.3]	19.1(11.5) [17.9, 20.3]
CS mm/s	325(164) [308, 342]	235(140) [222, 248]	290(164) [274, 307]	229(148) [214, 244]
CSS mm/s	143(76) [137, 149]	135(71) [129, 141]	149(79) [142, 156]	136(67) [130, 142]
CA °	115(15) [113, 117]	116(15) [114, 118]	115(16) [114, 118]	116(17) [113, 117]
CD mm	30.6(5.6) [30.0, 31.2]	38.3(5.9) [37.7, 38.9]	29.1(5.4) [28.6, 29.6]	26.8(5.5) [26.2, 27.4]

Table 3.1: Kinematic means with (STD) and [95% CI].

In summary, results generated from this formative study suggest that pin-and-cross actions can be completed within reasonable times with low error rates. Additional complexity introduced by a two-touch pin, or an extra drag before pin-and-cross, did have a profound effect on time and error, which proves both factors are viable variations. In the next section, results of this study is used to motivate design guidelines for pin-and-cross targets. Furthermore, kinematic measurements of this study are coded as recognition heuristics for pin-and-cross actions in the wild, and is successfully applied on an Android photo app for different usages, which we will discuss in Chapter 5.

3.3 Pin-and-Cross Menu Design Guidelines

Utilizing results generated from the previous formative study, now we are able to explore possible design guidelines to further optimize performance of pin-and-cross actions.

3.3.1 Target Angle

Admittedly, limited statistical significances were found for the effect of *Angle*, on time, error, and preference according the the formative study. One could even argue that all crossing target angles are equally suitable. However, given the large number of target positions we tested, the lack of significances is understandable. We believe the combined trends on time, error and preference by *Angle* showing in Figure 3.4 offers useful information on the design of pin-and-cross target, one important fact we observed is that some range of *Angle*s are more suitable than others under different task conditions (two-finger pins, or drag before pin). In order to justify the thresholds of "suitable" *Angle*s, we follow a set of criteria that consider the importance of time, error and preference all together. Using these criteria listed below, we are able to classify ranges of *Angle*s as more suitable (indicated in light yellow shading in Figure 3.4). Specifically, an *Angle* is considered as a suitable target if (Figure 3.4):

- The average performance time is not a on peak or plateau.
- Crossing error is less than a reasonable number 5% (dashed red line).
- Subjective preference is more than a neutral 3 (dashed green line).

These suitable angles are illustrated as optimum pin-and-cross layouts for one and two touch pin-and-cross menus, invoked after either a tap or a drag (Figure 3.6).

3.3.2 Target Length and Spacing

We used mean + 2`STD` of kinematic measurement *Crossing Distance (CD)* as the optimum length of crossing targets. Optimum spacing is determined from *Crossing Length (CL)*. Recommended values are labeled on Figure 3.6 as well.

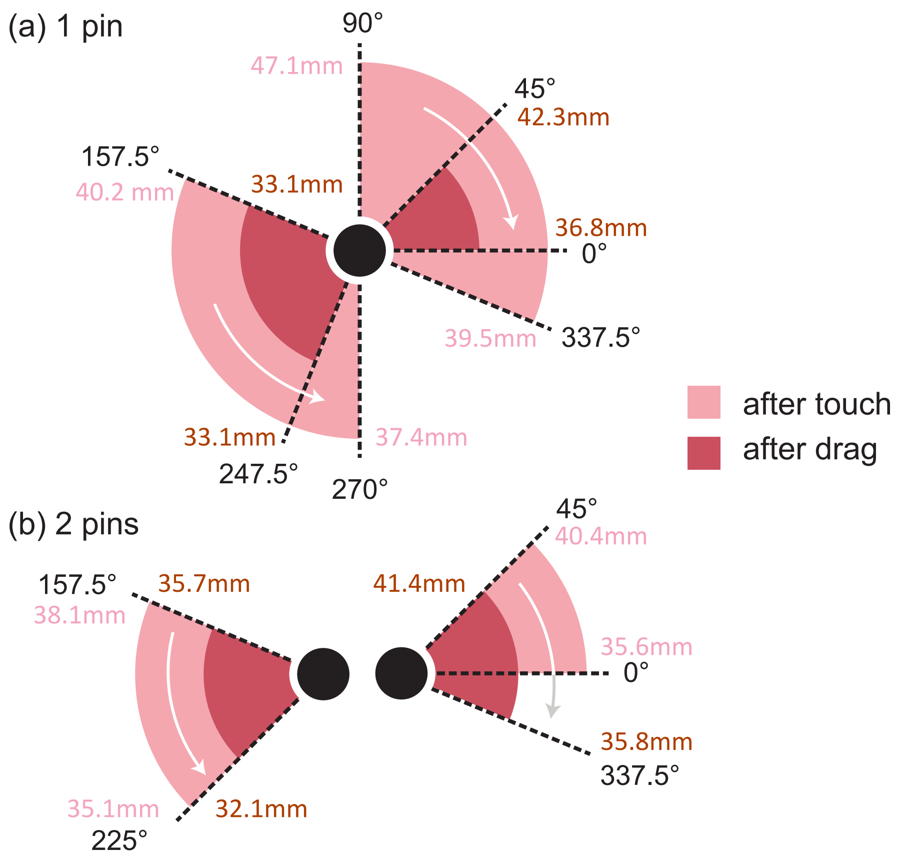


Figure 3.6: Optimum crossing target layout and crossing directions for pin-and-cross menus using: (a) 1 pin; (b) 2 pins. Arrows indicate preferred crossing direction.

3.3.3 Crossing Direction

A target can be crossed in two directions, clockwise or counter-clockwise, and trigger two different commands [3]. More frequently used commands should be triggered in the most natural direction. Using the logs from previous study, we calculated the ratio of clockwise to counter-clockwise crossing direction for each target Angle. Across all tasks, targets pointing right were typically crossed clockwise ($ratio > 0.5$ from $337.5°$ to $112.5°$) and targets pointing left were crossed counter clockwise ($ratio > 0.5$ from $135°$ to $270°$). Analysis on crossing direction ratios help determine which side of targets (below or above the segment line) should place more frequently selected menu items, as illustrated by the white arrows in Figure 3.6.

Based on the pattern of digits used for two touch pins, we suggest anchoring crossing targets off of each pin touch. For two touch pins, different subsets of suitable angles should be used, depending on whether it is expected to see thumb and index as pins (as in two touch transformations), or index and middle as pins (as in two touch scrolling). In practice, all crossing targets can be rotated with pin fingers to preserve the relative angles and positions of crossing target to pins.

3.4 Summary

In this chapter we conduct a formative study to evaluate the fundamental performance of four types of pin-and-cross variations: *1PIN-Touch*, *2PIN-Touch*, *1PIN-Drag*, *2PIN-Drag*. Two-finger pins, as well as drag movements before a typical pin-and-cross action was proved to have a profound impact on both task time and accuracy. Additionally, we analyze the effect of target angles on time, errors, preference, and observe consistent patterns over all pin-and-cross tasks.

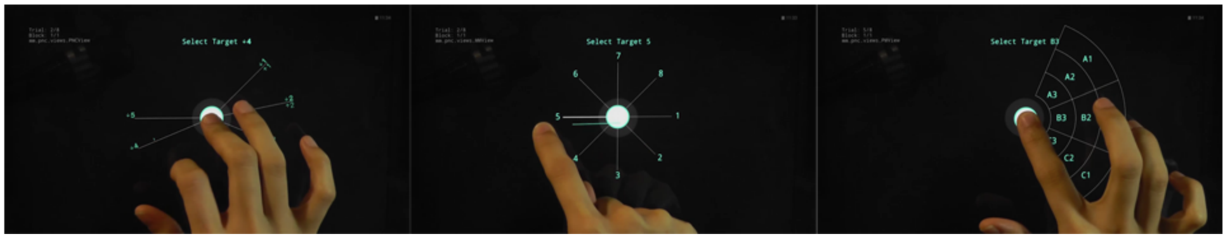
A set of kinematic characteristics calculated from the touch event logs reveal the natural behaviors on users performing pin-and-cross actions, which motivate the design of an optimum pin-and-cross menu layout. The layout design covers target length, spacing, positions, and crossing directions, and is suitable for menus that can be invoked with one or two pin fingers, either after a touch or a drag. Furthermore, analysis on kinematic input produces valuable design recommendations and guides the implementation of a pin-and-cross recognizer in Chapter 5.

Lastly, we notice that it is still necessary to compare pin-and-cross with other existing contextual menu designs, in order to demonstrate the applicability of pin-and-cross menu in practice, and unlock its potential advantages or disadvantages over traditional menu design. This motivates our second study in Chapter 4 where we compare pin-and-cross with a Marking Menu and partial Pie Menu.

Chapter 4

Comparative Study of Pin-and-Cross Contextual Menu

ADDITIONAL INFORMATION
This chapter is accompanied by a video which demonstrates the experiment setup of the comparative study described in this chapter.
The video is available at: http://youtu.be/NK-yRta38RA



To further explore potential advantages or disadvantages of pin-and-cross menu over traditional menu designs, we evaluate the performance of a one-pin-and-cross menu, along with a functionally equivalent Marking Menu and a partial Pie Menu. All menus support transition from novice mode to expert mode. Our study shows that pin-and-cross can be as accurate as existing menu designs and performs 27% faster when invoked on a draggable object.

4.1 Comparative Study

4.1.1 Experiment Design

The goal of this study is to compare a one-touch pin-and-cross context menu to functionally equivalent menu techniques: a single level marking menu [36] and tablet-adapted version of Banovic et al.'s partial Pie Menu [7]. The basic task is item selection on a contextual menu. To fully emulate the real world situation, where various manipulations can be operated at the same time (e.g. pinch to zoom, drag to transform, press-and-hold to switch mode), the task simulates a context menu on a movable object that requires a press-and-hold to disambiguate menu activation from object dragging for the marking menu and pie menu. All these menus support novices mode and expert mode as well.

Participants

Twelve right-handed participants were recruited (1 female), ranging in age from 22 to 29 years old. All reported experience with touch screen devices, and none of them participated in the formative study we discussed in Chapter 3.

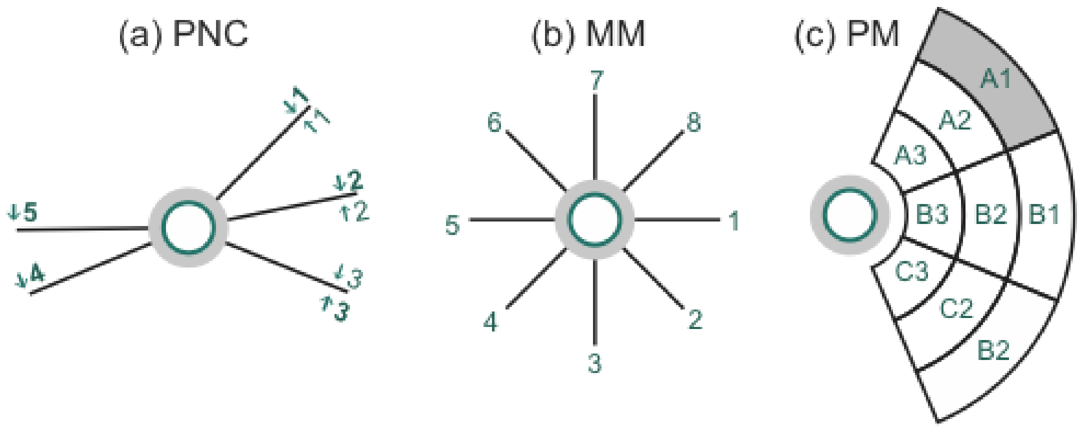


Figure 4.1: Menu techniques: (a) pin-and-cross (PNC); (b) marking menu (MM); (c) partial Pie Menu (PM).

Apparatus

The apparatus was identical to the set up of the formative study as described in section 3.2.1.

Tasks and Stimuli

Participants were instructed to activate a context menu on top of a movable object (rendered as a filled circle 13mm in diameter) and select a given menu item. The object could be translated with a one-touch drag, so menu activation had to be done without accidentally moving the object out of a central area (23mm diameter circle, gray shadow in Figure 4.1). This simulates using context menus on graphical objects like photo album thumbnails or drawing application shapes, where dragging, scrolling and taps can be operated without switching to other modes. For each *Technique*, target labels are chosen to be most salient (we piloted compass directions as labels, but participants found numbers more intuitive).

We tested three contextual menu techniques in this study:

- *Pin-and-Cross (PNC)*. We created a pin-and-cross menu for one finger pin using our design guidelines and implemented using our recognition heuristics (5.1) generated from kinematics data of the formative study. One digit pins the object while another digit on the same hand crosses through one of five line targets as shown in Figure 4.1a. Two targets on the left side could be crossed in a single direction, while the other three targets on the right side could be crossed in different directions, which results in eight possible item selections in total. Targets were labeled numerically with crossing direction icons (↑ 1, ↓ 1, ↑ 2, ↓ 2, etc.). We follow the design guidelines of pin-and-cross menu when choosing crossing target angles and lengths. All crossing targets are displayed after 150ms to reduce visual clutter for experts and encourage novice to expert transition [35].

- *Marking Menu (MM)*. We implemented an 8 item marking menu with targets placed at 8 principal compass directions (4.1b). This layout is suggested the optimum for a single level menu [36]. Target directions were indicated with 26mm lines and labeled with numbers clockwise arranged beginning from East (1, 2, 3, ...). To disambiguate the marking stroke from dragging the object, the finger had to press-and-hold the object for a short dwell time of 333 ms (as suggested by Kurtenbach & Buxton [35]) to activate the contextual menu. The menu was then displayed 150ms after activation to encourage novice users achieve expert transition. Selection started when the finger moved past the pin boundary after dwell time expired, and was confirmed after either by lifting the finger or drawing a mark longer than the 26mm line length.

- *Partial Pie Menu (PM)*. We implemented a partial pie menu adapted from Banovic et al. [7] where one touch opens the menu and another touch with the same hand selects a menu item on lift-off. Targets are distributed within three ring sectors, each ring contains three targets which creates 9 possible menu items in total. In order to be consistent with other *Technique*s, we only use 8 targets by excluding the one at upper right since this was slowest in Banovic et al.s study. Targets were labeled using a letter for sector and number for ring (A1, A2, A3, B1, ...). The inner ring of targets begins 8mm to the right of the touch position. Each ring is 15mm thick and targets have an arc angle of 22.5 (target sizes are approximately 15×18 to 15×40mm). Similar to Marking Menu, a 333ms press-and-hold period is required before activating the menu to disambiguate menu selection from dragging.

Task Errors and Time

A trial was considered to be successful if the correct menu item was selected.When any error occurred, a tone sounded and the same trial was restarted. Note that accidental dragging is a recoverable error, participants could drag the object back to the center before activating the menu. We record the number of errors, but only use the error-free trials in non-error rate analysis.

The total task time was from the first touch down until the item was selected with a cross, mark, or tap. We further divide task time into three *Activities*:

- *Waiting* is the total time that the first touch is still before performance, note that this actually includes dwell time, visual search to locate target position, and time to prepare for the real selection. We used the threshold from pin-and-cross heuristics recognizer to distinguish the moment of finger still and moving.
- *Dragging* is the total time that the first touch is moving. Drags can occur either before or after the activation of menu.
- *Performing* is the time spent doing the selection part of technique. For *PNC*, this is from the start of the crossing stroke until it intersects the line target. For *MM*, this is from the moment when the marking stroke passes over the pin boundary until the finger is lifted or the stroke length exceeds 26mm. For *PM*, this is from the moment when users touch down to select target until the finger lift up to confirm the selection.

Novice to Expert Transition

Kurtenbach & Buxton [35] discuss user behavior of novice and expert when using marking menu. Novices need to find out what commands are available on the given menu and how to invoke commands. Thus, allowing longer mental perception time should be considered when designing interfaces for novices. Experts, who are already aware of available commands, usually put speed first and desire a faster command invocation. A user's expertise increases gradually thus seamless transition between novice and expert mode should be supported. Press-and-hold is an option to achieve mode switch.

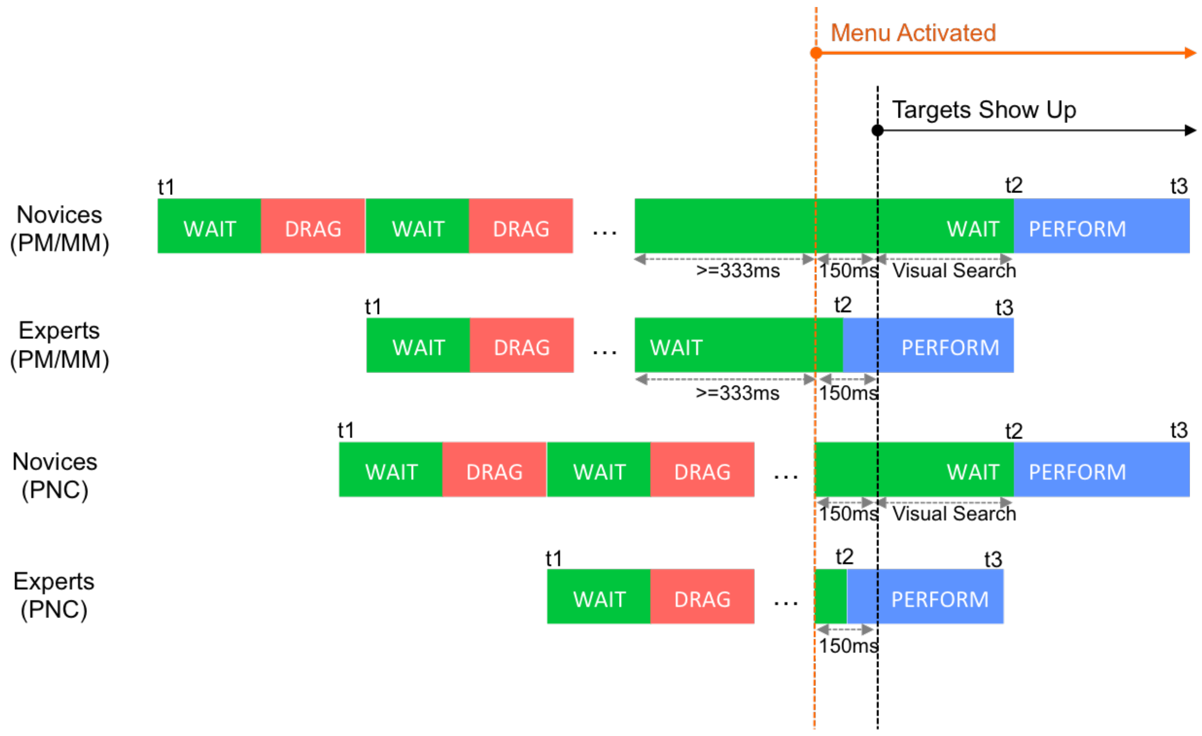


Figure 4.2: Typical behaviors of novices and experts on *PM/MM* and *PNC*

Novices hold longer to percept menu and experts start selection shortly after the press without much waiting.

In this study, we display menu targets after 150ms to reduce visual clutter for experts and encourage novice to expert transition. As illustrated in Figure 4.2, we use a 333ms press-and-hold time to activate a *PM* or *MM* contextual menu. After menu being activated, novices tend to hold until targets pop up, which leads to longer dwell time (green blocks). Experts who are familiar of available commands (i.e. location of targets) and how to invoke commands (e.g. tap a sector for *PM*, draw a mark for *MM*, cross a target clockwise or counter-clockwise for *PNC*) tend to start performing (blue blocks) immediately. They need less dwell time and they can even start selection before targets show up. Similarly, pin-and-cross experts tend to initialize crossing less than 150ms after the first touch. 333ms press-and-hold time is excluded because pin-and-cross naturally disambiguates from existing direct manipulations such as dragging. Note that *Dragging*s (red blocks) are possible for all tasks before the start of press-and-holding (for *PM/MM*), or pin (for *PNC*).

Design and Protocol

The experiment design is within-subjects, repeated measures, full factorial. ll trials for each *Technique* were presented together, with *Technique* order counterbalanced using a Latin Square. For each *Technique*, participants received a short instructional demo, then performed 4 blocks of measured trials (the first block is considered a practice block). Each block presented the 8 menu items in random order. Short rest breaks were enforced at the end of each block. In summary:

3 *Technique*s (*PNC*, *MM*, *PM*) $\times$
4 blocks $\times$
8 menu items $\times$
12 participants
= 1,152 data points

4.1.2 Results

All tests use within-subjects repeated-measures ANOVA with Holm correction for post hoc tests. We removed outlying data points with times more than 3 standard deviations from the *Technique* mean. This removed 2.08%, 1.82%, 0.50% (8, 7, 2 out of 384) for *PNC*, *MM*, and *PM* respectively. Although block 1 is considered practice, we included it when testing for learning effects. Testing each *Technique* separately, block had a significant effect on time for MM ($F_{3,33} = 5.583, p < 0.004$) and PNC ($F_{3,33} = 4.084, p < 0.02$). Post-hoc tests show block 2 is faster than block 1 ($p < 0.022$) for PNC, and block 3 is faster than block 1 ($p < 0.027$) for MM. For fair comparison, we only use blocks 3 and 4 to analyze all Techniques.

Time

There is a main effect of *Technique* on task time ($F_{2,22} = 13.69, p < 0.001$). Post hoc analysis shows *PNC* (801ms) is faster than both *MM* (1092ms) and *PM* (1107ms) (all $p < 10e^{-7}$) (Figure 4.3). To better understand the characteristics of each technique, we break the task time down into three periods: waiting, dragging, and performing. Performing the crossing selection action is very fast with *PNC*, however with a long waiting time. This reveals that motor or cognitive preparation is required for *PNC* selections. *MM* and *PM* have similar motor or cognitive preparation requirements, and are even longer due to the extra 333ms dwell time. We will discuss this in more details in the next section 4.1.3. The average dragging time of *MM* is very small (21ms), which suggests users minimal tendency to begin marking before the dwell period completes.

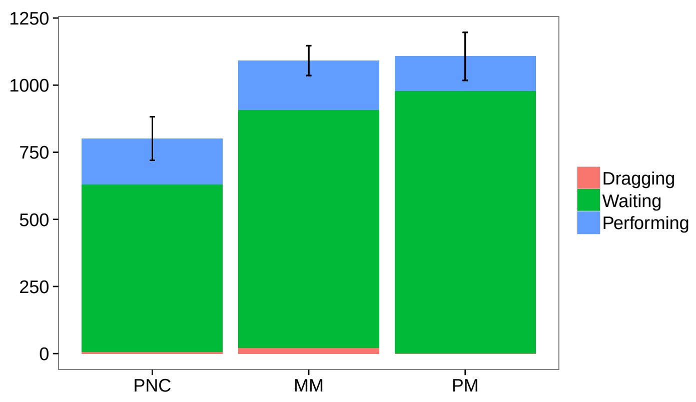


Figure 4.3: Time by *Technique*

Error

There is no main effect of Technique on error rate. Rates are reasonable good and similar (PNC 3.5%, MM 4.5%, PM 6.3%).

Preference

Participants were instructed to rate each *Technique* on a numeric scale from 1 to 5 (5 is highly preferred, 3 is neutral) at the end of the experiment. *MM* was rated highest (4.5), then *PNC* (3.17), then *PM* (3.10). There is a main effect on ratings by technique ($F_{2,22} = 6.54, p < 0.006$). Post-hoc analysis shows that *MM* is significantly higher than *PM* and *PNC* with however no difference is found between *PM* and *PNC*. Based on participant comments, the neutral rating for PNC is likely because it is quite different than all current multi-touch interactions.

4.1.3 Discussion

In this study all contextual menus we investigated were used on a movable object. Under this scenario, pin-and-cross is shown to be 27% faster than an functionally equivalent marking menu or a partial pie menu. Note that, under pin-and-cross selection, users do not need to wait for 333ms to enter in the a menu selection mode, which is certainly one of the main reasons of faster completion time. The built-in distinction between dragging and menu activation pin-and-cross has is a clear advantage over traditional menu designs.

To compare techniques that are suitable for implementations without any movable objects, we subtracted the 333ms dwell time from *MM* and *PM* and re-analyzed the task time. With this adjustment, we found no significant difference between the three techniques (p = 0.807). However, considering that a one-finger pin-and-cross menu can actually support more than 8 items by enabling selections on both directions for all crossing targets, and that pin-and-cross can be further combined with existing techniques such as marking menus (we will provide an example in the next chapter), we believe that pin-and-cross context menu is as good as or better than these previous techniques.

Lastly, similar to marking menus, pin-and-cross directly supports transitions from novice to expert mode. Novice become an expert after several selections where the same motor action is always used and the selection ultimately becomes more automatic and faster. We also examined the data to see how many participants were able to achieve the transition from a novice to an expert for each *Technique*. Specifically, we calculated the number of trials participants started performing the selection technique before the menu graphics appeared (less than 150ms after a pin touch finger or after dwell completed on the marking menu). We found 1 trial where these conditions held for pin-and-cross, 17 for the marking menu, and 38 trials for pie menu. This is not entirely discouraging considering the amount of practice required to attain expert performance with marking menus [29].

4.2 Summary

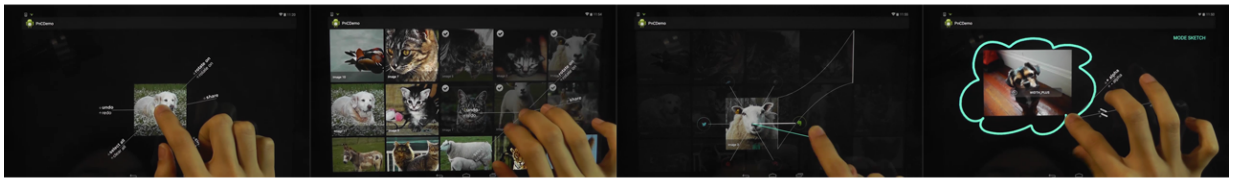
In this chapter, we compare a one-touch pin-and-cross context menu to a functionally equivalent marking menu [7] with single level, and a partial pie menu. Our results show that pin-and-cross is just as accurate and 27% faster when invoked on a draggable object. For implementations without draggable objects, pin-and-cross is as good as previous techniques, and even better considering that one crossing target can actually break down to two according to the crossing direction, and that pin-and-cross menu can be combined with previous technique together to achieve better performance.

We will demonstrate the applicability of pin-and-cross menu in the next chapter.

Chapter 5

Pin-and-Cross Recognition and Demonstration Applications

ADDITIONAL INFORMATION
This chapter is accompanied by a video which demonstrates the usage of pin-and-cross techniques described in this chapter.
The video is available at: http://youtu.be/NK-yRta38RA



Motivated from the kinematics analysis results characterized from our formative study in Chapter 3, we are able to produce a set up heuristics that can be used to build a pin-and-cross recognizer. In this chapter, we describe the implementation guidelines on a pin-and-cross recognizer, which is then used to create five different pin-and-cross techniques that are all presented by an Android photo app.

5.1 Recognition Heuristics of Pin-and-Cross

Compared to a machine learning classifier, which is commonly used by other previous work, heuristics-based recognizer is simple to create. Empirical thresholds can be directly applied on the recognizer and fast to execute. The experiment results from Chapter 3 are used for empirically-derived heuristics to recognize pin-and-cross actions distinct from other direct manipulation movements like dragging, scrolling, and two finger *rotate-scale-translate transformations* (RST-transformations). Seven rules are produced when testing the kinematic characteristics of pin touch(s) and the crossing stroke touch.

5.1.1 Empirically Derived Heuristics

Note that it is not easy to distinguish which finger is applied on a tablet environment, i.e. can not distinguish between index finger and middle finger without peripheral sensor such as a camera. For generality, we test all possible input digits and make no assumptions regarding touches intended as pin(s) or a cross. This means that all touch subsets with two or three fingers given a current touch event will be considered as a possible input digits for pin-and-cross selection. All permutations of pin and cross assignments will be tested. To further generalize our recognizer to be adaptable on *Touch* and *Drag* variations, we test all values that are calculated using a maximum time window of 20 frames (333ms). All thresholds are derived from *1PIN-Touch* kinematic statistics unless noted. There is little variation in kinematic statistics across tasks. *1PIN-Touch* is the most common type and most representative type of pin-and-cross, and all other variations can ultimately be decomposed to a series of *1PIN-Touch* tasks.

A pin-and-cross menu is shown if the following rules hold (refer Table 3.1 for all kinematics measurements):

- *P1*: Cumulative pin movement distance must be less than 2.8 mm (33 px). The threshold is *mean + 2STD* of the *Pin Distance (PD)* kinematic statistic.
- *P2*: Average pin velocity must be less than 16.9 mm/s (0.2 px/ms). The threshold is the upper 95% CI of the *Pin Speed (PS)* kinematic statistic.

Note that the above two rules also determine the number of pins (e.g. one or two touches can satisfy P1 and P2). Examining the current command mode (e.g. scrolling, or direct manipulation) and targets under pins (e.g. thumbnail), the arrangement of active crossing targets are determined and displayed. Then, as long as *P1* and *P2* continue to hold, the activated target will be selected if the following crossing rules also hold:

- *C1*: Average crossing stroke velocity must be greater than 308 mm/s (3.64 px/ms). The threshold is the lower 95% CI of the *Crossing Speed (CS)* statistic.

- *C2*: the instantaneous stroke velocity must be greater than 135 mm/s (1.6 px/ms). The threshold is the lower 95% CI of the *Crossing Start Speed (CSS)* statistic.
- *C3*: The angle of the crossing stroke relative to the vector from stroke to pin must be less than $132°$. The threshold is *mean+2STD* of the *Crossing angle (CA)* statistic.
- *C4*: No more than 792ms have passed since the crossing stroke started. The threshold is *mean+2STD* of the trial time across all tasks.
- *C5*: The crossing stroke intersects with a crossing target. In our later implementation, we found that *C5* can be relaxed by defining multiple virtual crossing targets near a single visible target. We did this for a downward target after scrolling (described in section 5.2.2).

These heuristics were developed and tested with dragging, one and two-finger scrolling, and two-finger RST transformations on an Android photo app, which we will demonstrate on the next section.

5.2 Demonstration Applications of Pin-and-Cross

To demonstrate the applicability of pin-and-cross menu and its recognizer, we created a tablet photo app embedded with four pin-and-cross techniques. We implemented a one-finger contextual menu enabling smooth activation on edge objects, and can be combined with marking menu to achieve multi-level menu selections. Traditional two finger scrolling was extended to support faster performance. Pin-and-cross could also be used to change direct manipulation modes seamlessly, and toggle constrains with transformations such as rotation, scaling and translation.

5.2.1 One Finger Pin-and-Cross Contextual Menu

We implemented a one-finger pin-and-cross contextual menu that supports a set of operations performed on a thumbnail such as select, edit, rotate, copy, cut, and delete (Figure 5.1). The menu has five crossing targets where three are on the right side while others on the left. Target positions follow our design guidelines and is very similar to the menu we used in the comparison study as in Chapter 4). The menu provides access to 10 contextual operations. For example, command "rotate image CW/CCW" are on either side of the upper-right crossing target and can be accessed using an index finger pin and middle finger cross (Figure 5.1a); a multiple object selection mode can be entered and exited using a lower left crossing target; "copy" and "cut" are assigned to a lower right crossing target and can be accessed with an index pin and thumb cross;

"undo/redo" are on either side of a left crossing target and accessed with a middle finger pin and index cross.

Pin-and-cross menu can also be combined with other existing multi-touch techniques. Crossing item "share" immediately activates an 8-item marking menu for the pin finger (Figure 5.1b). Without lifting any finger, the pin finger moves and makes the mark to select an item (e.g. select "share on Evernote" as shown in the screenshot).

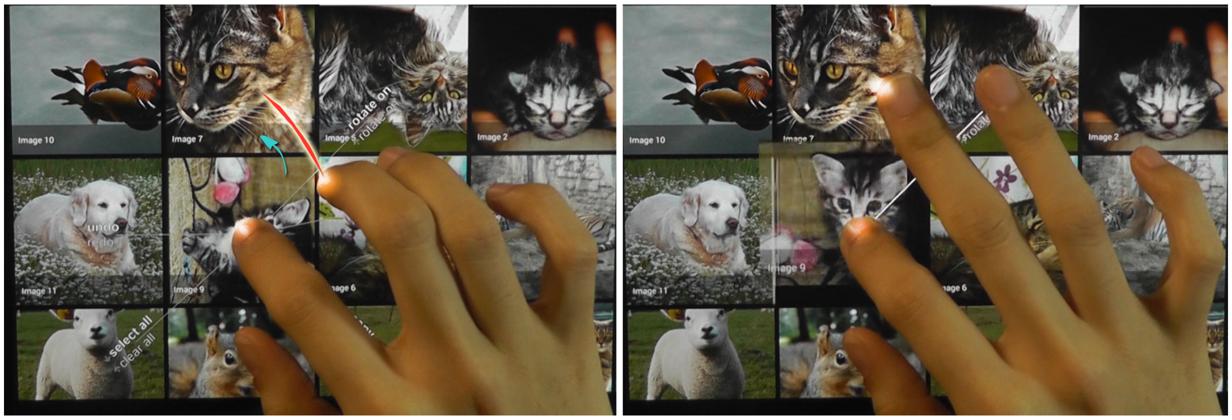
Pin-and-cross, like other existing context menu such as marking menu and pie menu, requires sufficient space around the contextual object in order to trigger the command. Objects arranged on the corner would have problems displaying some menu items (menu activated on the upper right corner cannot display the right most targets). To remedy this issue and support activation of pin-and-cross on objects near the edge, we use a spring back technique where the corner object can be dragged out of corner and placed in the center, but after a cross is detected and the selection is completed, releasing the pin snaps the thumbnail back to its original location (Figure 5.1c).

5.2.2 Extending Two Finger Scrolling and Panning

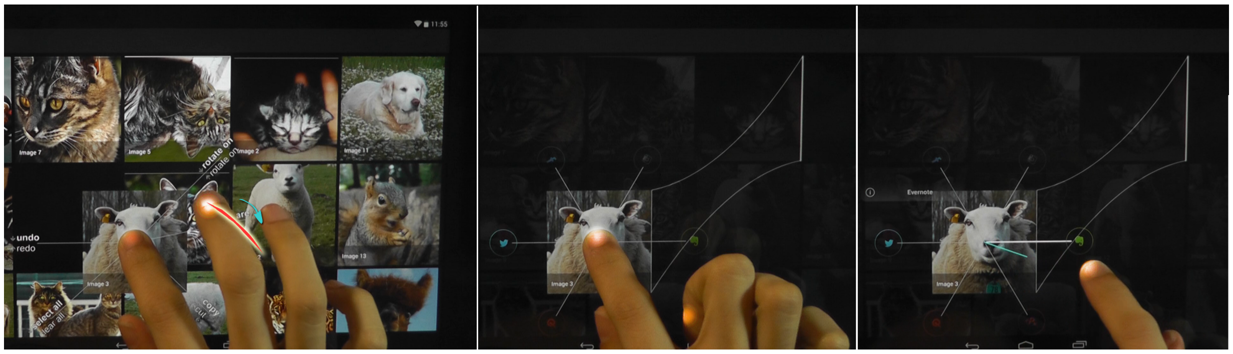
The app uses two-finger scrolling to disambiguate from single finger thumbnail dragging. We also implement a pin-and-cross menu with scrolling acceleration commands at the end of each scroll gesture. Since scrolling typically uses index and middle, or middle and ring, a target pointing to south is anchored off the index finger with command "go to end" and command "go to top" on either side and can be crossed with the thumb (as illustrated in the right most screenshots of Figure 5.2). A target pointing to the east is anchored off the middle finger, command "page down" and command "page up" are placed on either side of the target and can be crossed with the ring finger (see left and middle screenshots of Figure 5.2).

5.2.3 Constraining Two-Dimensional Object Transformation

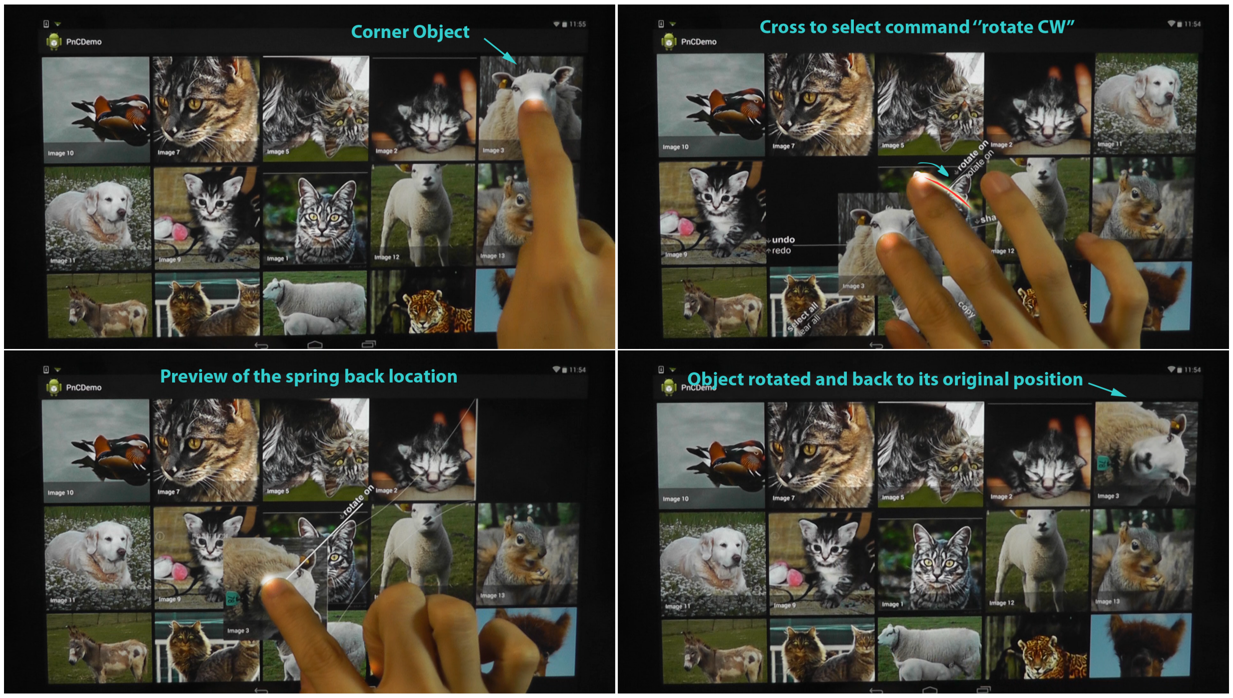
The app implements a collage where one or more photos can be added using a standard two touch (usually two digits from index, middle, or thumb) RST transformation. A two-finger pin-and-cross menu is created on the collage to toggle transformation constraints during photo manipulation. For example, after activating/deactivating a right sided target with two-finger pin-and-cross (pin with thumb and index, cross with middle), we turn on/off a $45°$ angle snapping while rotating, thus the image being manipulated can only rotate by the multiple of $45°$, as shown in Figure 5.3. This two-finger pin-and-cross menu can be triggered immediately without extra dwell time. By using our pin-and-cross recognition heuristics, we are able to distinguish the two-finger pin-and-cross action with other common two-finger RST transformations.



(a) index finger pin, middle cross to access command "rotate CW"



(b) contextual menu combined with marking menu


Corner Object
Cross to select command "rotate CW"
Preview of the spring back location
Object rotated and back to its original position
PnCDemo

(c) spring back technique for performing contextual menu on edge objects

Figure 5.1: One-finger pin-and-cross contextual menu

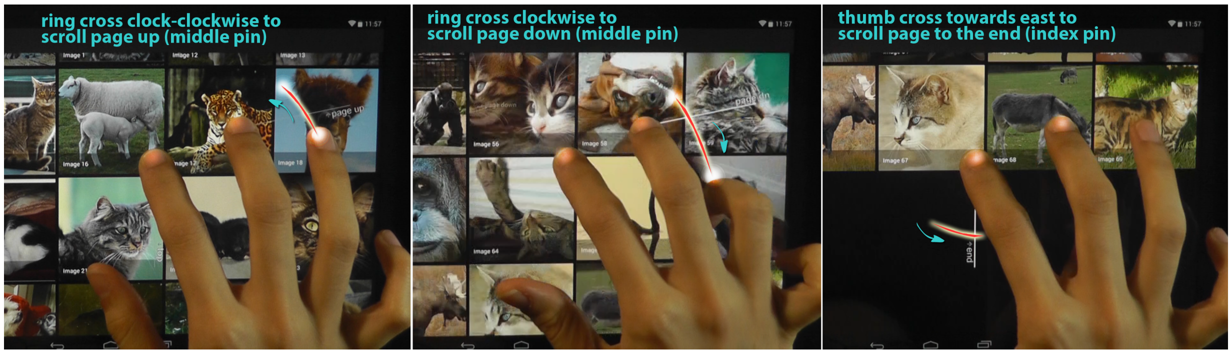

ring cross clock-clockwise to scroll page up (middle pin)
page up
ring cross clockwise to scroll page down (middle pin)
page down
thumb cross towards east to scroll page to the end (index pin)
end

Figure 5.2: Extending two finger scrolling

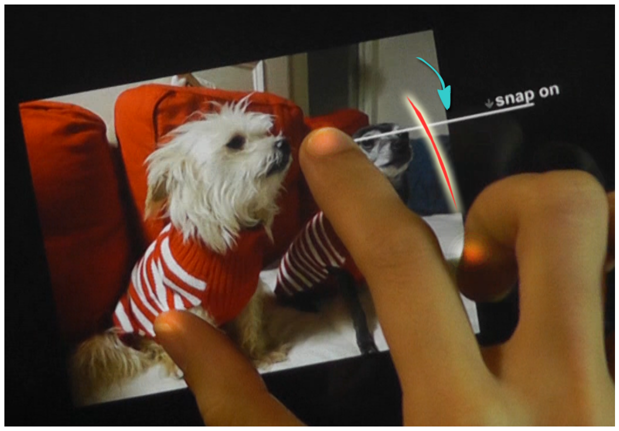
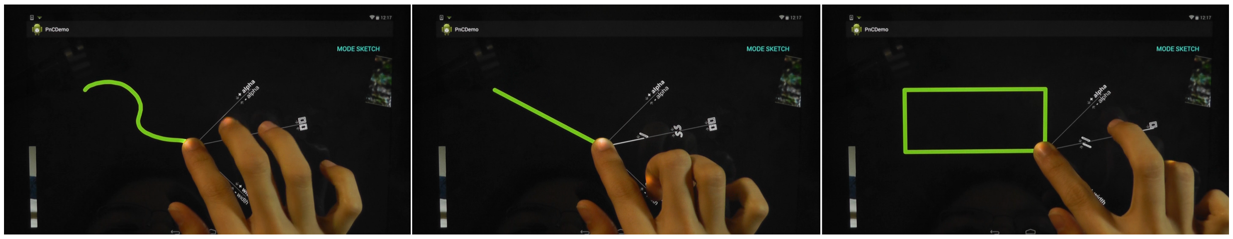


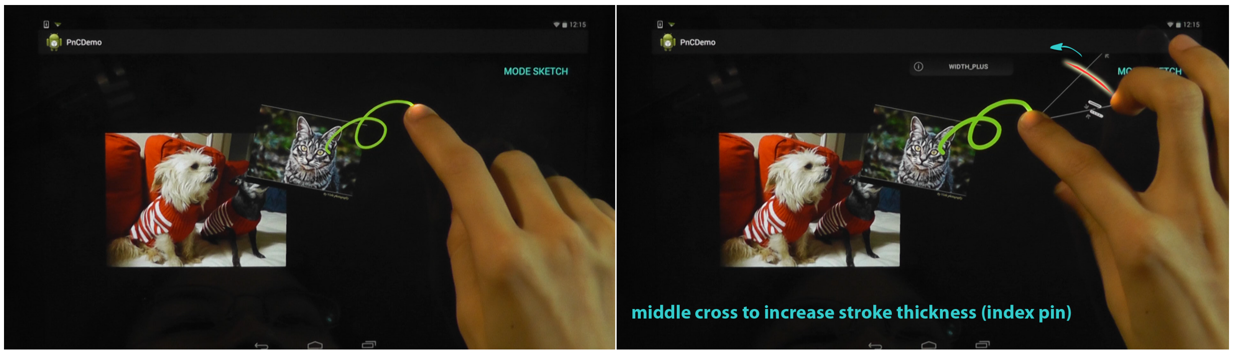
Figure 5.3: Constraining object transformation

5.2.4 Direct Manipulation Mode and Attribute Selection

Users are free to add any annotations to the collage, or any photos placed on the collage. A two-finger pin-and-cross menu us used to switch to the annotation mode. Various drawing tools are provided under the annotation mode. For example, by using a one-finger pin-and-cross menu, users can change the drawing mode to freehand, straight line, or rectangle without lifting the index finger while drawing, as illustrated in Figure 5.4a. Additionally, drawing attributes such as stroke width, stroke transparency, and color, can all be adjusted without interrupting the current drawing stroke as seen if Figure 5.4b. Note that crossing targets are segmented where three different modes or attributes are placed on one crossing target, which bring in a new variation of pin-and-cross menu to be further explored.



(a) changing drawing mode from curve, to line, to rectangle without lifting drawing finger



(b) changing the value of drawing attibute without lift drawing finger

Figure 5.4: Pin-and-cross performed under direct manipulation mode

5.3 Discussion

These recognition heuristics were developed and tested with dragging, one- and two-finger scrolling, and two-finger RST transformations on the Android photo app we described in this chapter. In practice, we noticed there is one infrequent rotation action that can be detected as pin-and-cross accidentally: one finger is stationary as a pivot while another finger rotates the object. However, we found most people do not rotate like this, and people will modify their rotation style to avoid false activation.

Additionally, our heuristics are mostly produced from *1PIN-Touch* kinematics statistics, although we found it worked well even for two-fingered pins, either after taps or after a drag, multiple recognizers could be built using the same heuristics however utilizing task specific kinematics values given in table 3.1 (e.g. two-finger pin-and-cross menu generates heuristic thresholds from task *2PIN-Touch*).

Furthermore, quantitative tests could be added to test the disambiguation ability of pin-and-cross technique and its recognizer. A log study that collects users' natural touch interactions on tablet could be potentially useful in exploring patterns pin-and-cross shares with other actions, and on determining false positive rates of the recognizer. Specifically, participants would be given a laboratory tablet to play with for half an hour. To simulate real world scenario, participants would use the tablet just as usual. The laboratory tablet installs some commonly used applications (e.g. photo app, sketch app, drawing app, etc.) that require touch manipulations, and a touch event logger to collect all touch events produced by users. We can then run our recognizer and calculate the number of pin-and-cross actions naturally performed by users during their daily usage. This would help us determine false positive recognitions, and better understand how pin-and-cross distinguishes from existing touch operations. The study could also be a chronological experiment in order to collect larger data set, e.g. participants come every other day during a certain period of time, i.e. a week.

Chapter 6

Conclusions

6.1 Summary

This thesis evaluates and models an interaction technique called crossing selection. The performance of touch crossing is validated with an in-depth empirical investigation on six types of touch crossing tasks, and is demonstrated a new application of FFitts law. Results of our controlled experiment on crossing performance provide evidence to support, and guidelines to evaluate more crossing-based interaction technique in touch interfaces.

With touch crossing being validated, we then developed, evaluated and demonstrated a new multi-touch interaction space called "pin-and-cross", where crossing is combined with static touches to achieve higher expressibility. To better understand the characteristics of pin-and-cross, we conducted two evaluation experiments.

The first evaluation was an empirical experiment where we explored four types of pin-and-cross variations regarding time, error, subjective preferences, input kinematics, and target layouts. Results of this study are served as design guidelines for a optimum pin-and-cross menu, and build a heuristic-based recognizer that can be used on one or two finger pin-and-cross menu after a drag or just static touch(es).

The second evaluation was a comparative study where we compare a one-finger pin-and-cross contextual menu with functional equivalent marking menu and a partial pie menu, and found that pin-and-cross is as accurate and 27% faster when invoked on a draggable object. As our comparative evaluation showed, a big advantage for pin-and-cross is the ability to "pin" an object and issue a contextual command without any press-and-hold.

Lastly, we implemented a photo app to demonstrate more pin-and-cross variations such as two-finger accelerated scrolling, drawing modes or attributes changing while drawing, two-finger constrained transformations, and pin-and-cross combined with a Marking Menu.

6.2 Future Work

6.2.1 More Expressive Multi-Touch Crossing

With touch crossing empirically validated as a practical and efficient selection technique, we can now explore more novel forms of touch crossing like pin-and-cross.

For example, we can assign different actions according to the number of fingers used to cross (Figure 6.1a); we can also utilize the geometric relationship between finger touches and the crossing target (e.g. the line segment between two touch points can be orthogonal or collinear with the crossing target, which will trigger different commands as shown in Figure 6.1b). Similar to pin-and-cross, multiple touch points cross a target from the same side can access different commands, however this time we use a crossing bundle instead of static touches. For example, one can place thumb and index finger on either side of a target, make a snapping movement while maintaining contact, and create a simultaneous crossing from both sides (Figure 6.1c). This movement can be further differentiated when performed in the reverse directions. A similar action can be made with the index and the middle fingers.

Combining touch input with crossing may provide a way to further increase touch expressibility and help close the performance gap when abandoning mouse and keyboard.

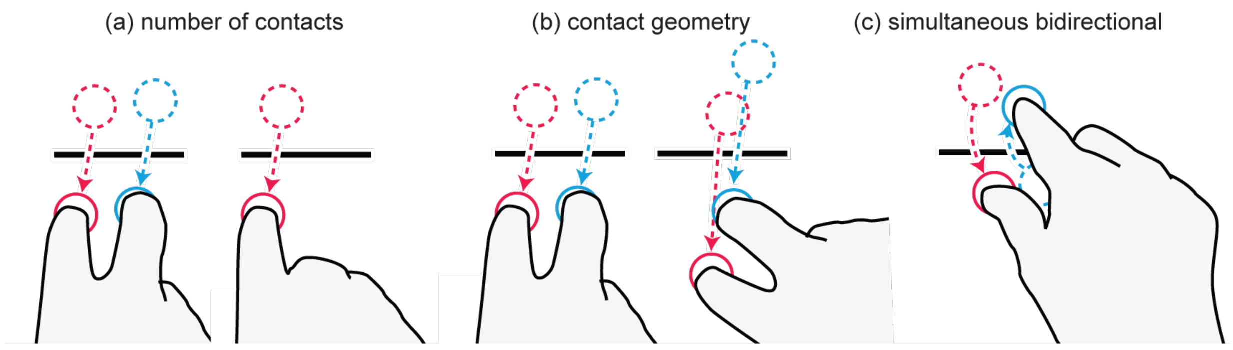


Figure 6.1: Examples of expressive multi-touch crossing: (a) different crossing actions depending on number of contacts; (b) different crossing actions by changing the crossing contact geometry; (c) simultaneous bidirectional crossing of two contacts similar to a “finger snap”

6.2.2 Pin-and-Cross: Challenges and Opportunities

Notice that pin-and-cross is not using exactly the same muscle groups as touch crossing due to the additional effort of performing “pinning”. Further investigation can be conducted to evaluate the motor speed of pin-and-cross style interactions.

Additionally, there is still a large space to be explored given the high versatility of pin-and-cross. For example, in our implementation, pin-and-cross could be used as a delimiter to invoke a marking menu. This suggests that pin-and-cross gesture could replace the traditional press-and-hold as a faster universal delimiter to activate other types of contextual menus in multi-touch interfaces.

Another area to explore is variation on the crossing target. For example, a typical crossing target can be segmented to display more items on a single line. We demonstrated a simple usage segmented crossing target at section 5.2.4, where three drawing attributes were placed on one crossing target. Study on the number of segments, segment size and segment spacing could be useful to unlock more possibilities behind pin-and-cross.

There are also more variations in the crossing action itself. Targets can be bundled together and be selected by just one crossing stroke. More crossing fingers can be added. For example, thumb is used to pin and then a nearby target can be crossed with different combinations of index, middle, and ring fingers to access up to 7 different commands similar to finger count menus [6]. Also, a static pin can be combined with an amplitude crossing, i.e. crossing movements towards or away from the pin finger trigger different commands, in order to further expand the vocabulary of pin-and-cross interactions.

Lastly, pin-and-cross focuses on one-handed interactions, but two-handed pin-and-cross is certainly worth exploring. One possible extension with two hands, which adapts the idea of consecutive distant taps [26], is to introduce consecutive distant crossings where a second cross occurs immediately after pinning.

6.3 Final Remarks

This thesis contributes to the understanding and creation of human computer interactions in touch input modalities.

We present an in-depth investigation on crossing paradigm, a smooth, seamless, and intuitive approach to achieve goal selection. We have explored and empirically validated fundamental crossing performance, and proved its applicability on touch input devices. Crossing can even be combined with multi-touch to further increase the expressibility of touch input space, and help close the performance gap when abandoning mouse and keyboard (e.g. enable shortcuts on touch display). For example, pin-and-cross, an interactive technique we have defined, explored and demonstrated in this work, shows how efficient, expressive and remarkable crossing could be when combining with static touches.

Moreover, though our experiments are conducted on tablet environment, the versatility and adaptability of crossing interaction, as well as pin-and-cross technique suggest enormous potentials on mobile applications. We believe that results and insights gained from this research would be a great asset to the exploration of innovative interfaces.

Appendices

A Pin-and-cross Leanring Effect

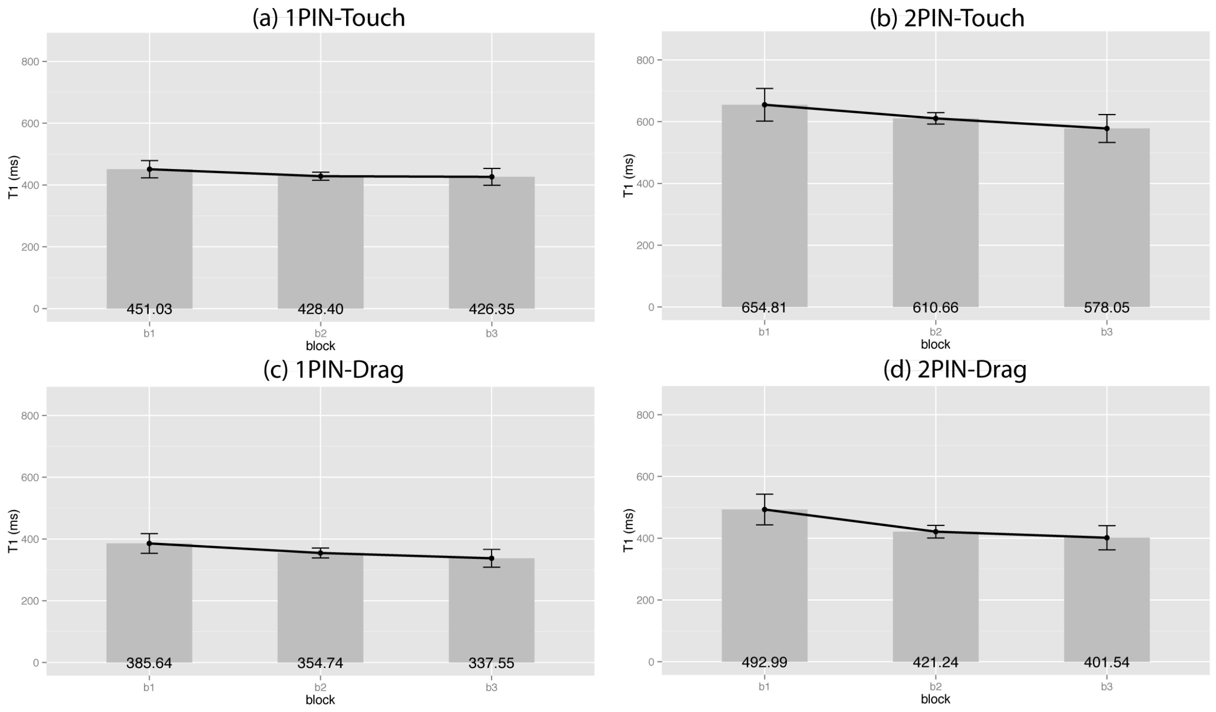


Figure A.1: Leanring effect of four pin-and-cross tasks

B Pin-and-cross Kinematic Values Distribution

All plots included in this section label values as described below:

- **Black** solid line: label mean value
- **Red** dashed line: mean + 2STD
- **Blue** dot-dashed line: lower 95% confidence interval

Please refer to Section 3.2.2 for more details on the following kinematic measurements.

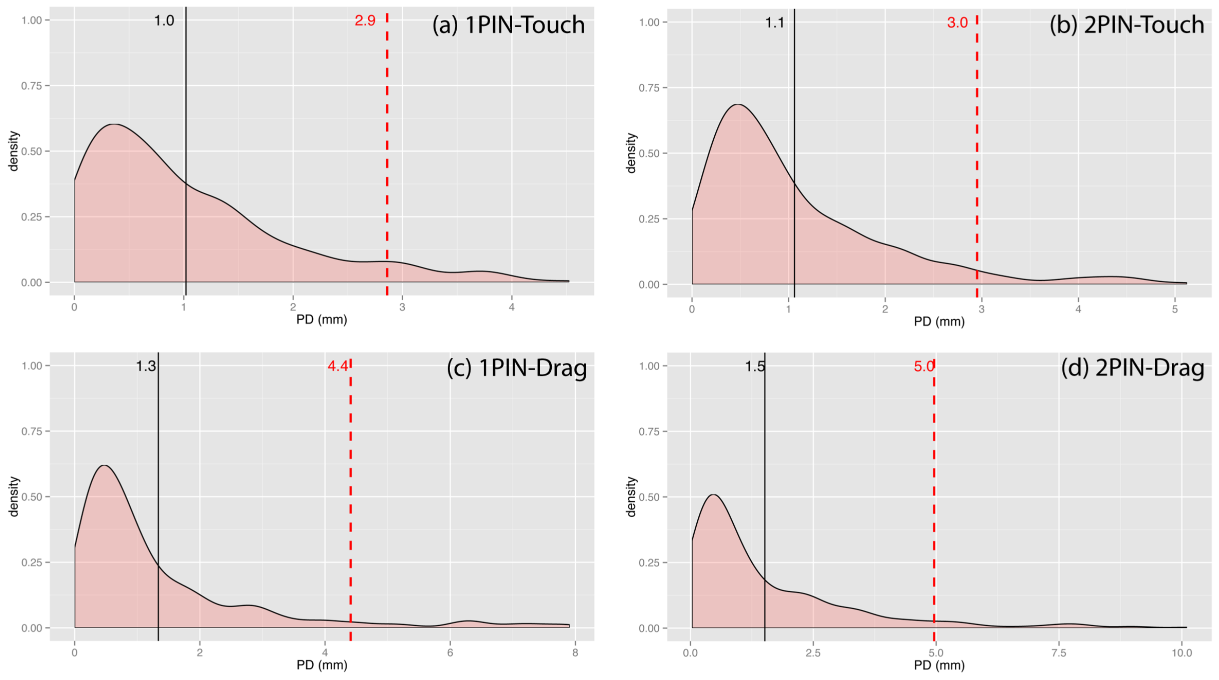


Figure B.1: Distribution of *Pin Distance (PD)*

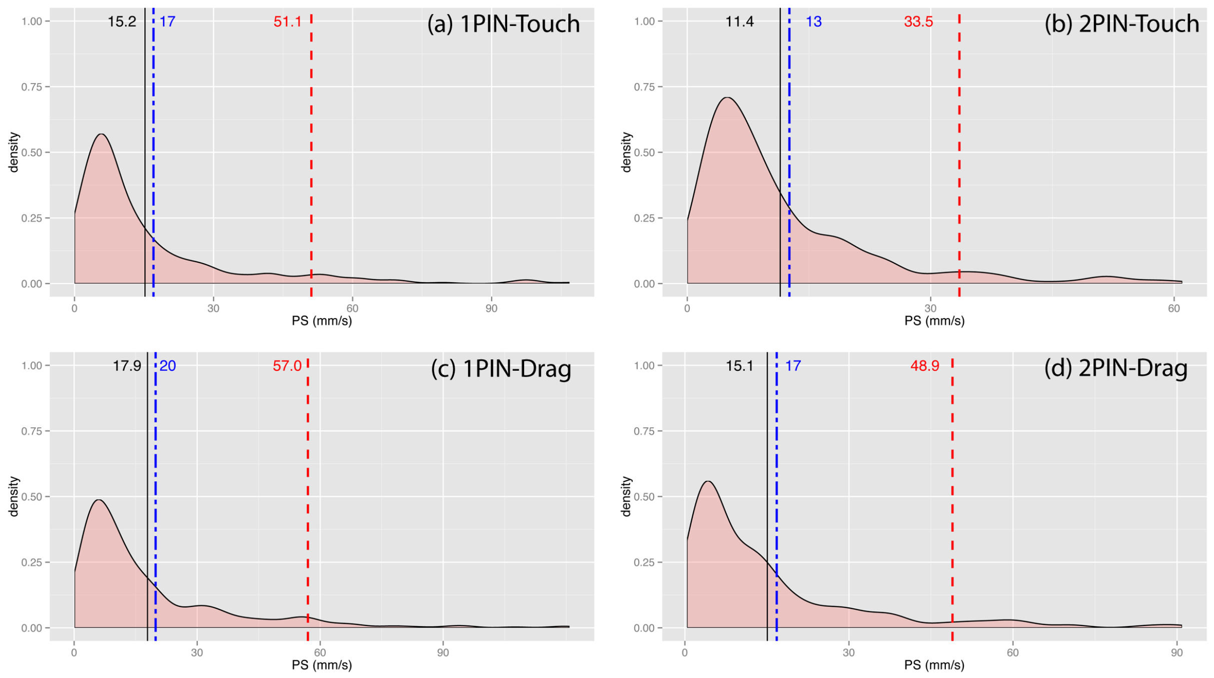


Figure B.2: Distribution of *Pin Speed (PS)*

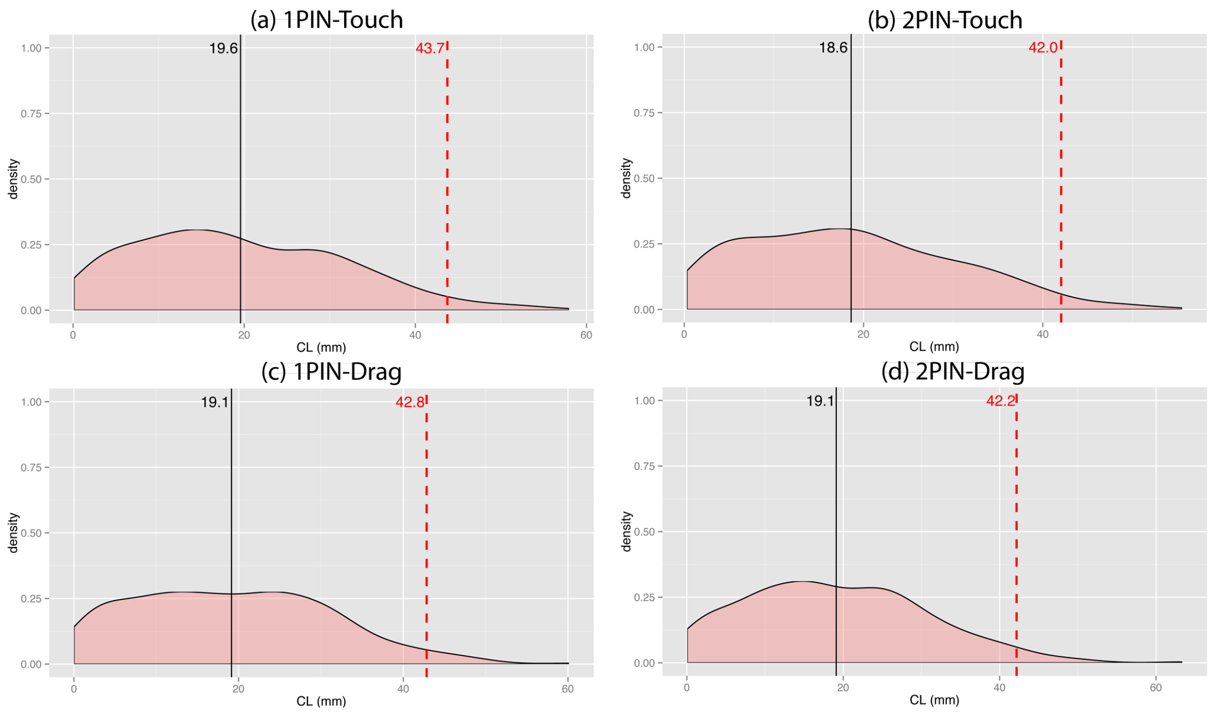


Figure B.3: Distribution of *Crossing Length (CL)*

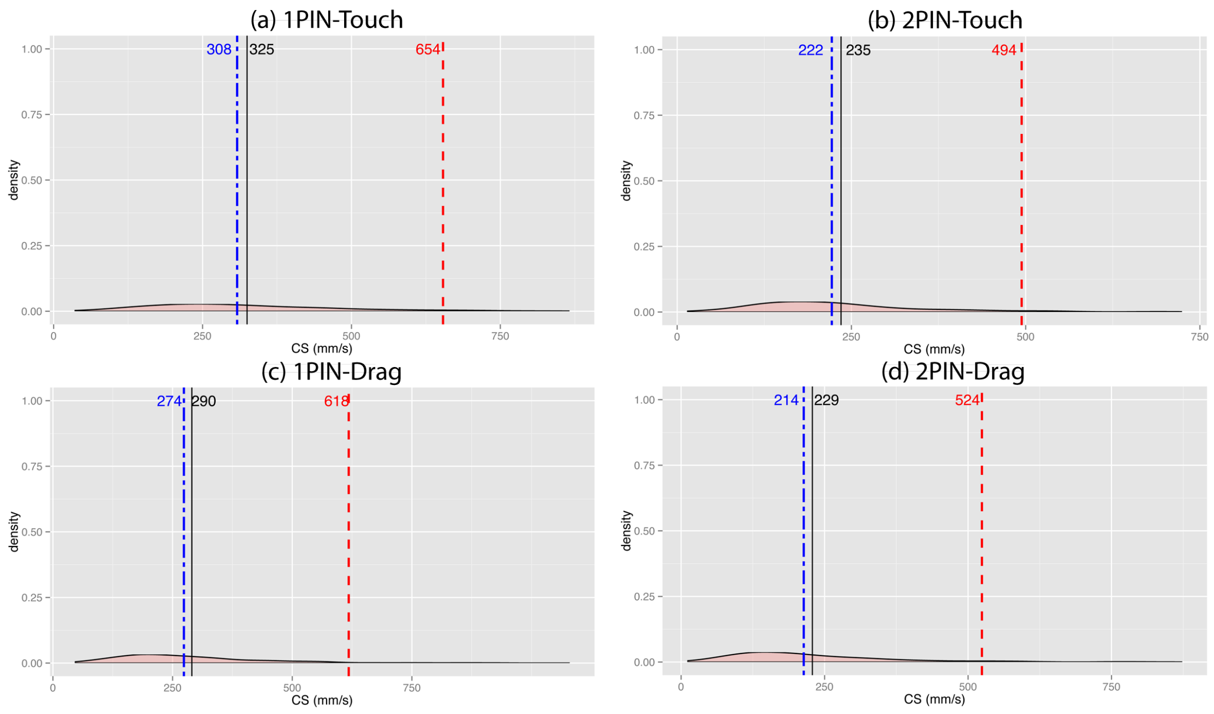


Figure B.4: Distribution of *Crossing Speed (CS)*

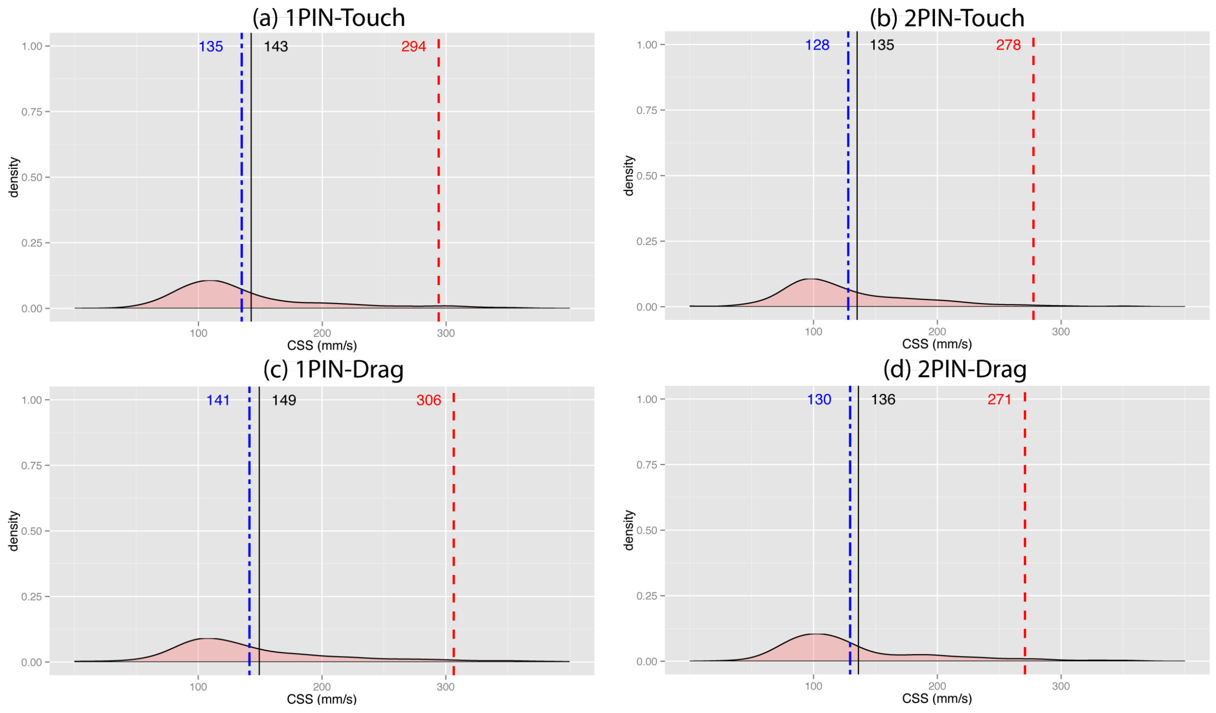


Figure B.5: Distribution of *Crossing Start Speed (CSS)*

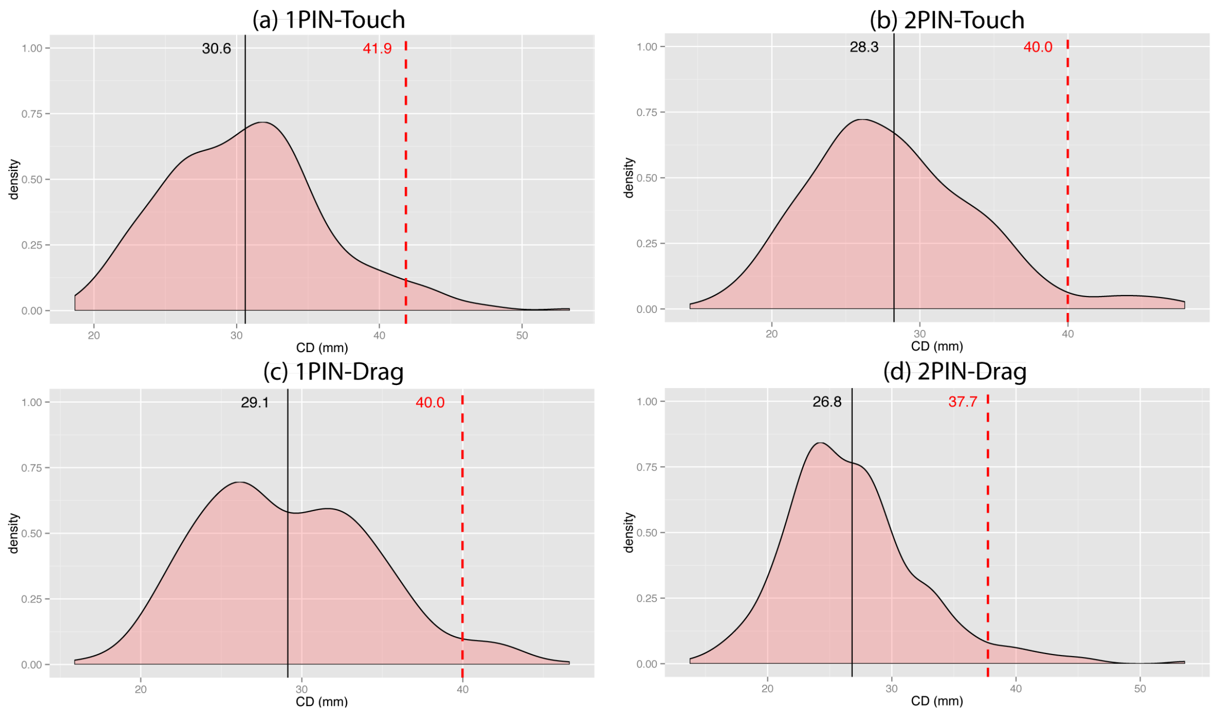


Figure B.6: Distribution of *Crossing Distance (CD)*

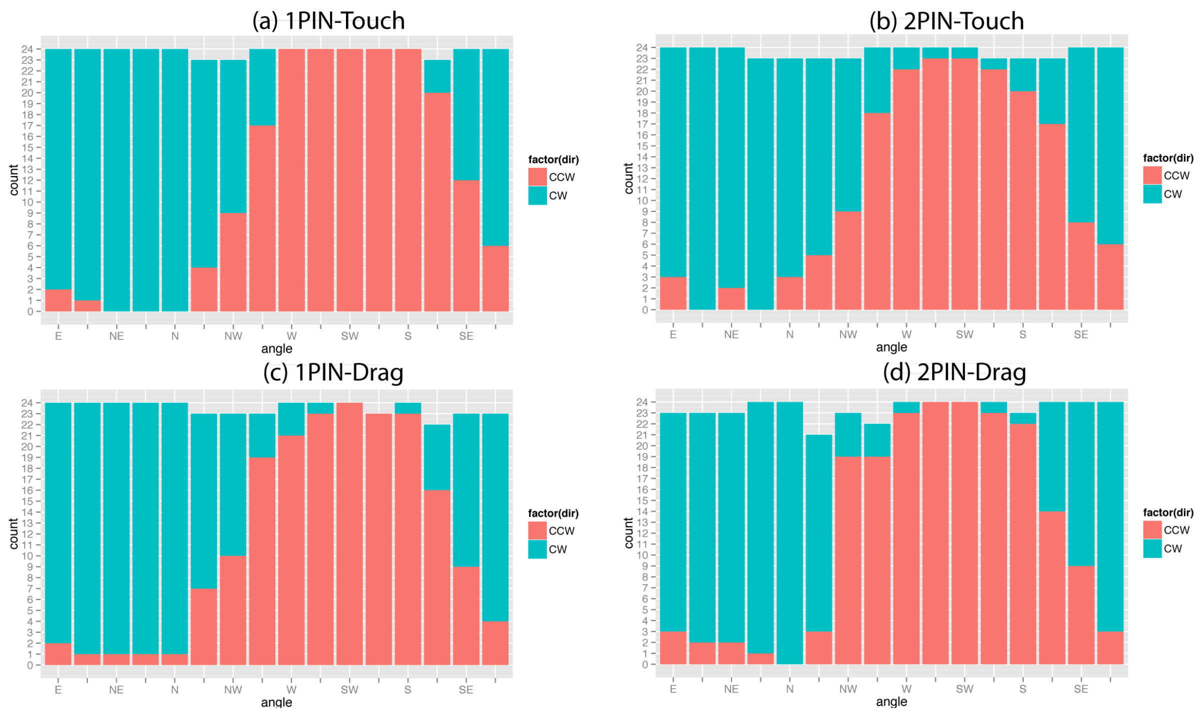


Figure B.7: Distribution of *Crossing Directions (CDir)* on *Angle*

References

[1] Johnny Accot and Shumin Zhai. Beyond fitts' law: Models for trajectory-based HCI tasks. In *Human Factors in Computing Systems, CHI '97 Conference Proceedings, Atlanta, Georgia, USA, March 22-27, 1997.*, pages 295–302, 1997.

[2] Johnny Accot and Shumin Zhai. Scale effects in steering law tasks. In *Proceedings of the CHI 2001 Conference on Human Factors in Computing Systems, Seattle, WA, USA, March 31 - April 5, 2001.*, pages 1–8, 2001.

[3] Johnny Accot and Shumin Zhai. More than dotting the i's - foundations for crossing-based interfaces. In *Proceedings of the CHI 2002 Conference on Human Factors in Computing Systems: Changing our World, Changing ourselves, Minneapolis, Minnesota, USA, April 20-25, 2002.*, pages 73–80, 2002.

[4] Georg Apitz and François Guimbretière. Crossy: a crossing-based drawing application. *ACM Trans. Graph.*, 24(3):930, 2005.

[5] Georg Apitz, François Guimbretière, and Shumin Zhai. Foundations for designing and evaluating user interfaces based on the crossing paradigm. *ACM Trans. Comput.-Hum. Interact.*, 17(2), 2010.

[6] Gilles Bailly, JöRg MüLler, and Eric Lecolinet. Design and evaluation of finger-count interaction: Combining multitouch gestures and menus. *Int. J. Hum.-Comput. Stud.*, 70(10):673–689, October 2012.

[7] Nikola Banovic, Frank Chun Yat Li, David Dearman, Koji Yatani, and Khai N. Truong. Design of unimanual multi-finger pie menu interaction. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ITS '11, pages 120–129, New York, NY, USA, 2011. ACM.

[8] Nikola Banovic, Frank Chun Yat Li, David Dearman, Koji Yatani, and Khai N. Truong. Design of unimanual multi-finger pie menu interaction. In *ACM International Conference on Interactive Tabletops and Surfaces, ITS 2011, Kobe, Japan, November 13-16, 2011*, pages 120–129, 2011.

[9] Michel Beaudouin-Lafon. Instrumental interaction: An interaction model for designing post-wimp user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '00, pages 446–453, New York, NY, USA, 2000. ACM.

[10] Hrvoje Benko and Daniel Wigdor. Imprecision, inaccuracy, and frustration: The tale of touch input. In *Tabletops - Horizontal Interactive Displays*, pages 249–275. 2010.

[11] Hrvoje Benko, Andrew D. Wilson, and Patrick Baudisch. Precise selection techniques for multi-touch screens. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 1263–1272, New York, NY, USA, 2006. ACM.

[12] Margrit Betke, James Gips, and Peter Fleming. The camera mouse: Visual tracking of body features to provide computer access for people with severe disabilities.

[13] Xiaojun Bi, Yang Li, and Shumin Zhai. Ffitts law: modeling finger touch with fitts' law. In *2013 ACM SIGCHI Conference on Human Factors in Computing Systems, CHI '13, Paris, France, April 27 - May 2, 2013*, pages 1363–1372, 2013.

[14] William Buxton, Ralph Hill, and Peter Rowley. Issues and techniques in touch-sensitive tablet input. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '85, pages 215–224, New York, NY, USA, 1985. ACM.

[15] A. Cockburn, D. Ahlström, and C. Gutwin. Understanding performance in touch selections: Tap, drag and radial pointing drag with finger, stylus and mouse. *Int. J. Hum.-Comput. Stud.*, 70(3):218–233, March 2012.

[16] Morgan Dixon, François Guimbretière, and Nicholas Chen. Optimal parameters for efficient crossing-based dialog boxes. In *Proceedings of the 2008 Conference on Human Factors in Computing Systems, CHI 2008, 2008, Florence, Italy, April 5-10, 2008*, pages 1623–1632, 2008.

[17] Pierre Dragicevic. Combining crossing-based and paper-based interaction paradigms for dragging and dropping between overlapping windows. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*, UIST '04, pages 193–196, New York, NY, USA, 2004. ACM.

[18] Wenxin Feng, Ming Chen, and Margrit Betke. Target reverse crossing: a selection method for camera-based mouse-replacement systems. In *Proceedings of the 7th International Conference on PErvasive Technologies Related to Assistive Environments, PETRA 2014, Island of Rhodes, Greece, May 27 - 30, 2014*, pages 39:1–39:4, 2014.

[19] Clifton Forlines and Ravin Balakrishnan. Evaluating tactile feedback and direct vs. indirect stylus input in pointing and crossing selection tasks. In *Proceedings of the 2008 Conference on Human Factors in Computing Systems, CHI 2008, 2008, Florence, Italy, April 5-10, 2008*, pages 1563–1572, 2008.

[20] Clifton Forlines, Daniel Wigdor, Chia Shen, and Ravin Balakrishnan. Direct-touch vs. mouse input for tabletop displays. In *Proceedings of the 2007 Conference on Human Factors in Computing Systems, CHI 2007, San Jose, California, USA, April 28 - May 3, 2007*, pages 647–656, 2007.

[21] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts, 1994.

[22] Tiago João Guerreiro, Hugo Nicolau, Joaquim A. Jorge, and Daniel Gonçalves. Towards accessible touch interfaces. In *Proceedings of the 12th International ACM SIGACCESS Conference on Computers and Accessibility, ASSETS 2010, Orlando, FL, USA, October 25 - 27, 2010*, pages 19–26, 2010.

[23] Carl Gutwin, Andy Cockburn, Joey Scarr, Sylvain Malacria, and Scott C. Olson. Faster command selection on tablets with fasttap. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14, pages 2617–2626, New York, NY, USA, 2014. ACM.

[24] Chris Harrison and Scott Hudson. Using shear as a supplemental two-dimensional input channel for rich touchscreen interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 3149–3152, New York, NY, USA, 2012. ACM.

[25] Chris Harrison, Julia Schwarz, and Scott E. Hudson. Tapsense: Enhancing finger interaction on touch surfaces. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 627–636, New York, NY, USA, 2011. ACM.

[26] Seongkook Heo, Jiseong Gu, and Geehyuk Lee. Expanding touch input vocabulary by using consecutive distant taps. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14, pages 2597–2606, New York, NY, USA, 2014. ACM.

[27] Christian Holz and Patrick Baudisch. The generalized perceived input point model and how to double touch accuracy by extracting fingerprints. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 581–590, New York, NY, USA, 2010. ACM.

[28] Christian Holz and Patrick Baudisch. Understanding touch. In *Proceedings of the International Conference on Human Factors in Computing Systems, CHI 2011, Vancouver, BC, Canada, May 7-12, 2011*, pages 2501–2510, 2011.

[29] Mohit Jain and Ravin Balakrishnan. User learning and performance with bezel menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 2221–2230, New York, NY, USA, 2012. ACM.

[30] Matt Jones, Philippe A. Palanque, Albrecht Schmidt, and Tovi Grossman, editors. *CHI Conference on Human Factors in Computing Systems, CHI'14, Toronto, ON, Canada - April 26 - May 01, 2014*. ACM, 2014.

[31] Shaun K. Kane, Jeffrey P. Bigham, and Jacob O. Wobbrock. Slide rule: Making mobile touch screens accessible to blind people using multi-touch interaction techniques. In *Proceedings of the 10th International ACM SIGACCESS Conference on Computers and Accessibility*, Assets '08, pages 73–80, New York, NY, USA, 2008. ACM.

[32] Matthew Kay, Kyle Rector, Sunny Consolvo, Ben Greenstein, Jacob O. Wobbrock, Nathaniel F. Watson, and Julie A. Kientz. Pvt-touch: Adapting a reaction time test for touchscreen devices. In *7th International Conference on Pervasive Computing Technologies for Healthcare and Workshops, PervasiveHealth 2013, Venice, Italy, May 5-8, 2013*, pages 248–251, 2013.

[33] Kenrick Kin, Björn Hartmann, and Maneesh Agrawala. Two-handed marking menus for multitouch devices. *ACM Trans. Comput.-Hum. Interact.*, 18(3):16:1–16:23, August 2011.

[34] Donald Knuth. *The TEXbook*. Addison-Wesley, Reading, Massachusetts, 1986.

[35] Gordon Kurtenbach and William Buxton. User learning and performance with marking menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '94, pages 258–264, New York, NY, USA, 1994. ACM.

[36] Gordon Paul Kurtenbach. *The design and evaluation of marking menus*. PhD thesis, University of Toronto, 1993.

[37] Leslie Lamport. *LATEX — A Document Preparation System*. Addison-Wesley, Reading, Massachusetts, second edition, 1994.

[38] G. Julian Lepinski, Tovi Grossman, and George W. Fitzmaurice. The design and evaluation of multitouch marking menus. In *Proceedings of the 28th International Conference on Human Factors in Computing Systems, CHI 2010, Atlanta, Georgia, USA, April 10-15, 2010*, pages 2233–2242, 2010.

[39] Yuexing Luo and Daniel Vogel. Crossing-based selection with direct touch input. In *CHI Conference on Human Factors in Computing Systems, CHI'14, Toronto, ON, Canada - April 26 - May 01, 2014*, pages 2627–2636, 2014.

[40] I. Scott MacKenzie. Fitts' law as a research and design tool in human-computer interaction. *Human-Computer Interaction*, 7(1):91–139, 1992.

[41] Takashi Nakamura, Shin Takahashi, and Jiro Tanaka. Double-crossing: A new interaction technique for hand gesture interfaces. In *Proceedings of the 8th Asia-Pacific Conference on Computer-Human Interaction*, APCHI '08, pages 292–300, Berlin, Heidelberg, 2008. Springer-Verlag.

[42] Charles Perin, Pierre Dragicevic, and Jean-Daniel Fekete. Revisiting bertin matrices: New interactions for crafting tabular visualizations. *IEEE Trans. Vis. Comput. Graph.*, 20(12):2082–2091, 2014.

[43] Volker Roth and Thea Turner. Bezel swipe: Conflict-free scrolling and multiple selection on mobile touch screen devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 1523–1526, New York, NY, USA, 2009. ACM.

[44] Marcos Serrano, Eric Lecolinet, and Yves Guiard. Bezel-tap gestures: Quick activation of commands from sleep mode on tablets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 3027–3036, New York, NY, USA, 2013. ACM.

[45] Craig Stewart, Michael Rohs, Sven Kratz, and Georg Essl. Characteristics of pressure-based input for mobile devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 801–810, New York, NY, USA, 2010. ACM.

[46] Ahmed N. Sulaiman and Patrick Olivier. Attribute gates. In *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology, Monterey, CA, USA, October 19-22, 2008*, pages 57–66, 2008.

[47] Felix Thalmann, Ulrich von Zadow, Marcel Heckel, and Raimund Dachselt. X-o arch menu: Combining precise positioning with efficient menu selection on touch devices. In *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces*, ITS '14, pages 317–322, New York, NY, USA, 2014. ACM.

[48] Daniel Vogel and Patrick Baudisch. Shift: A technique for operating pen-based interfaces using touch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, pages 657–666, New York, NY, USA, 2007. ACM.

[49] Julie Wagner, Stéphane Huot, and Wendy Mackay. Bitouch and bipad: Designing bimanual interaction for hand-held tablets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 2317–2326, New York, NY, USA, 2012. ACM.

[50] Daniel Wigdor, Hrvoje Benko, John Pella, Jarrod Lombardo, and Sarah Williams. Rock &#38; rails: Extending multi-touch interactions with shape gestures to enable precise spatial manipulations. In *Proceedings of the SIGCHI Conference on Human*

Factors in Computing Systems, CHI '11, pages 1581–1590, New York, NY, USA, 2011. ACM.

[51] Jacob O. Wobbrock and Krzysztof Z. Gajos. Goal crossing with mice and trackballs for people with motor impairments: Performance, submovements, and design directions. *TACCESS*, 1(1), 2008.

[52] Jacob O. Wobbrock, Meredith Ringel Morris, and Andrew D. Wilson. User-defined gestures for surface computing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 1083–1092, New York, NY, USA, 2009. ACM.

[53] Mike Wu and Ravin Balakrishnan. Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. In *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology*, UIST '03, pages 193–202, New York, NY, USA, 2003. ACM.

[54] Takuto Yoshikawa, Buntarou Shizuki, and Jiro Tanaka. Handywidgets: local widgets pulled-out from hands. In *Interactive Tabletops and Surfaces, ITS'12, Cambridge/Boston, MA, USA, November 11-14, 2012*, pages 197–200, 2012.

[55] Chuang-Wen You, Yung-Huan Hsieh, and Wen-Huang Cheng. Attachedshock: Facilitating moving targets acquisition on augmented reality devices using goal-crossing actions. In *Proceedings of the 20th ACM International Conference on Multimedia*, MM '12, pages 1141–1144, New York, NY, USA, 2012. ACM.