HybridPointing Touch: A Technique to Switch Between Absolute and Relative Pointing on Large Touch Screens

by

Terence Dickson

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, 2017

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

This thesis includes first-authored peer-reviewed material that has been submitted to appear in the conference proceedings published by the Association for Computing Machinery (ACM). The conference paper from which I have adapted content is the following:

Terence Dickson, Rina Wehbe, Fabrice Matulic, and Daniel Vogel. HybridPointing Touch: A Technique to Switch Between Absolute and Relative Pointing on Large Touch Screens. Submitted for publication in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18. New York, NY, USA, 2018. ACM.

Abstract

We extend Forlines et al.'s idea of mixed absolute and relative "HybridPointing" to large multitouch displays. In our bimanual "CursorTap" interaction technique, one hand triggers a kinaesthetic relative pointing mode while the other controls a distant cursor similar to a large touchpad. A controlled experiment compares CursorTap to standard absolute touch input, and a "Drag" technique, where all display content may be dragged to the user. Our results show CursorTap is fastest when accessing distant targets and returning to nearby targets, a common usage scenario. Overall, the measured selection times across distances is nearly flat for CursorTap, but linearly increases for the absolute and Drag techniques. As further validation, a second study explores how CursorTap is used in a more open setting. We developed a two-person game to create opportunities to use CursorTap under cooperative and competitive settings. The results demonstrate a willingness to use CursorTap, and reveal situations and input modalities where usage is most likely.

Acknowledgements

I'm very grateful for the help of my advisor Dan Vogel, whose hard work in mentoring me made this thesis possible, and whose understanding toward my health needs went beyond what I had ever experienced in a professional setting. I also extend thanks to Rina Wehbe and Fabrice Matulic for their assistance in experiment design, equipment setup, and running experiment participants, especially when I could not have done so alone.

I also extend my gratitude to Craig Kaplan, Chrysanne DiMarco, Ed Lank, Christopher Batty and Jeff Avery, with whom I enjoyed curricular and extracurricular projects through my degree, allowing me to expand my research in meaningful and fascinating ways.

Special thanks to Edith Law, Dan Brown, Gladimir Baranoski and Yuying Li, whose help during my undergraduate degree encouraged and supported me in getting my Master's degree; as well as Olga Zorin and Dave Tompkins, with whom I had an incredible time helping to teach new students entering the world of Computer Science.

Finally, I'd like to acknowledge my family; my parents, Eva and Tim Dickson, my sister Julia Dickson, and my love Elijah Slaughter, whose support has been uncompromising through all obstacles.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Touch input almost always uses *absolute direct input*, where the finger manipulates graphical objects by touching them directly. While absolute direct input is well suited to touch screens like phones, tablets, and tabletops, it can be problematic on very large wall-sized displays. Not all parts of a large display may be easily within reach, the user may have to stretch their arms, twist their waist, stand on their toes, or walk a great distance to interact with areas outside a comfortable "arms-length" range. With multiple users, people may have to reach across each other to access content, which can feel intrusive [1] and their body occludes parts of the display for other users. In addition, there are issues related to parallax [17], front-projection shadows, and friction when dragging (often exacerbated by bezels or gaps between tiled displays) that can reduce the effectiveness of absolute direct touch input.

Forlines et al. [7] proposed using *relative indirect input* with a pen on a large wall-sized display to address many of the issues identified above. They recognized that users would wish to switch between absolute direct and relative indirect modes, so they designed a *HybridPointing* technique for pens. Since pens have limited input capabilities, they use something called a trailing widget to perform the mode switch. A quick and accurate movement of the pen "traps" a nearby floating button and the pen is switched into relative indirect input until it is lifted away from the display. This method proved effective for the pen, but it required accurate hover sensing which is not common with touch screens, and with multitouch, there is a larger input space available which could simplify the absolute-to-relative mode switch.

We extend Forlines et al.'s idea of mixed absolute and relative HybridPointing to very large multitouch displays. In our bimanual *CursorTap* interaction technique, one hand

triggers a kinaesthetic relative pointing mode while the other controls a distant cursor in the manner of a large touchpad. A controlled experiment compares CursorTap to standard absolute touch input, and a *Drag* technique, where all display content may be dragged close to the user. Our results show CursorTap is fastest when accessing distant targets and returning again to nearby targets, a common large display usage scenario. Overall, measured selection times across distances is nearly flat for CursorTap, but linearly increases for the absolute baseline and Drag. As a further validation, we also conduct a study exploring how people use CursorTap in a more realistic setting. We developed a two-person game that provides opportunities to use the technique under cooperative and competitive settings. The results demonstrate a willingness to use CursorTap, and reveal situations where usage is most likely.

1.1 Contributions

This thesis makes the following contributions:

- A novel input method for use on large touchscreens that allows the use of a gesture to switch modes between absolute touch input and a movable cursor, enabling long-distance multi-user interaction
- An implementation of this input method on top of a bare multi-touch layer, supporting multiple simultaneous users
- A set of filtering operations which can be applied to the input received from large touchscreen hardware to mitigate several classes of errors in current technology
- An implementation of these filtering methods on top of a bare multi-touch layer, used beneath all input methods tested
- An experiment comparing the cursor technique against existing absolute and relative touch techniques in both accuracy and speed
- An exploratory experiment with multiple simultaneous users demonstrating the properties of the hybrid cursor technique for a variety of real-world input modes

Chapter 2

Related Work

We survey work comparing absolute and relative input, interaction techniques for touch displays, as well as multi-user considerations. Our focus is on touch input on large vertical displays, but other devices and input are discussed if relevant.

2.1 Comparisons between Absolute and Relative Input

Forlines et al. [7] introduced and investigated the HybridPointing mode-switching technique that allows both absolute and relative pointing using a pen-based large wall interface, upon which HybridPointing Touch is based. We follow their experiment structure, with modifications made to accommodate differences in our technique and the touch screen context. We exploit multitouch for switching rather than hover sensing which is not typically available for touch. Ballagas et al. [2] compares several methods of smartphone input for both regular-sized and large screens, both absolute and relative. Among these methods, several factors are compared: Cognitive load, Perceptual load, Visual acquisition, Motor load, Motor acquisition, Ease of learning, Fatigue, Error proneness, and Distance sensitivity. While relative interaction methods varied in the fatigue placed upon the user, absolute interaction methods were all mid-to-high fatigue. However, absolute interaction methods all ranked mid-to-low on cognitive load, while almost all of the relative input techniques ranked mid-to-high. This supports the notion that absolute and relative input are well-suited to different tasks, and having the ability to switch between the two may improve user experience.

Kim et al. [11] designed interaction techniques for smartphones and other small touchscreens that allow bringing targets closer to the user's thumb while holding the

device in one hand. Their thumb-lever technique multiplies the user's thumb movements, acting similarly to a very simplified version of our *CursorTap* technique, without support for interaction. Their mode-switching gestures are tailored for small smartphones. Though little performance improvements were observed across different mode-switching operations and interaction techniques relative to the baseline, they found significant user preference for their novel methods.

2.2 Interaction Techniques for Large Touch Displays

Pen Input

Pen input is associated with a sitting posture on a horizontal surface, and most prior work on digital pens reflects this.

Baudisch et al. [4] implement two methods, called Drag-and-Pop and Drag-and-Pick, which handle the specific scenario of dragging an object along a large touchscreen to another distant target which is compatible with the dragged target. By creating local copies of the distant targets, the need to interact at a distance is eliminated. They observed notable improvements in selection time at long distances using these techniques.

Reetz et al. [16] proposes a method for moving targets along a touch or pen table surface. The technique involves flicking targets quickly, creating an intuitive motion with low accuracy. Hovering pen input allows the user to move the object after it is flicked away, supplementing accuracy. Participants enjoyed the physicality of the gesture, and were able to quickly move to distant targets.

Parker et al. [15] implemented a method for interacting with distant targets on a pen table by simply pointing toward the target. This could be used to perform interactions at a distance, or to pull the target toward the user to be interacted with using absolute pen input. Large and distant targets were faster to select using this technique.

Pen input includes another state when not in contact with the screen (hover mode), but loses out on the ability to perform multi-touch gestures.

Mid-air and Touch

Given the standing posture and large display context, previous research has explored how distant targets can be acquired using mid-air gestures, sometimes in combination with touch.

Jakobsen et al. [9] analyzed a combination of touch and mid-air gestures to select targets on large touch screens. The study found participants would choose to use mid-air gestures to select distant targets as the amount of effort required to move over to the target increased, but that these gestures did not provide any performance increase and were typically slower.

Pointable [3] implemented long-distance interaction on touch screens using a ray-casting technique, where interactions at a distance were performed at the intersection between the screen and a ray from the bridge of the user's nose through their fingertip. The long-distance interaction method did not demonstrate any performance improvement over direct touch, but the authors did not break down their results based on target distance.

Gunslinger [12] provided an implementation of mid-air gestures performed at the user's sides. The use of Gunslinger to select distant targets was proposed, however it is was only tested in the context of a user standing far from the display.

Our focus is a purely touch-based implementation of mode switching and distant interaction, since mid-air gestures require sensing not typically available on a large display.

Entirely Touch

There are prior works that propose methods for acquiring distant targets on large touch screens in a multi-user setting, without additional hardware.

Su et al. [18] propose a technique which is applicable to large touch screens, wherein a user may "jump" from target-to-target by inputting a movement direction. Though interesting, the technique is specifically tailored to situations with very small, dense targets, which may not be realistic for a large touch screen setup. The technique is not tested on large touch screens, and is also not compared to a baseline of absolute touch, instead being compared to other relative input techniques for navigating dense target setups.

WallPad [8] addresses long-distance interaction on large touch interfaces using what is effectively a touchpad near the user. They manipulate a cursor by dragging inside a defined rectangle, and click on targets by pressing a large rendered button. To switch to this form of relative input, they use a two-handed gesture where one finger slides "off" the end of the other finger. This simultaneously switches to absolute mode and defines an initial position for the cursor on the display using a ray cast along the gesture

trajectory. Using a literal touchpad has the advantage of familiarity for users who are used to working with laptops, and multi-touch extensions can support touchpad gestures such as pinch-to-zoom. However, the gesture appears somewhat complex, and the technique was not empirically tested. It remains unclear if limiting the relative input space and requiring taps on specific buttons is needed for usability, considering these constraints may limit performance. Note that their demonstrations are more focused on creating many Wallpads at the same time, to semi-permanently activate remote widgets. In contrast, our focus is on the most efficient method to switch between absolute and relative input.

Frisbee [10] creates a local magnifying glass that shows a copy of a distant area of the screen, which enables long-distance interaction through direct touch on the local copy, and allows for controls such as magnification of the copied area. This technique showed significant benefits in selection time for distant targets. Unlike our hybrid technique, movement of the frisbee itself requires the use of multiple controls to actually move both the target area and its representation in front of the user, and also is a single persistent widget as opposed to a mode-switch operation that resets to the user's position when the mode is entered.

We did not compare against these kinds of widget-based interaction techniques, as they are typically more specialized for a particular task and require more infrastructure to implement in an arbitrary application, compared to HybridTouch Pointing. The Frisbee technique does not support lasso selection without further development, instead opting to explore specialized tasks such as zoom, auto-panning, and display rotation. Magnifying the screen within the Frisbee also requires access to either the application or the display framebuffer, and cannot simply be implemented as an overlay.

Chapter 3

Relative Touch Techniques

In this section we describe our CursorTap technique, as well as the simple Drag baseline technique.

CursorTap

Our focus is on a HybridPointing technique, where the user can switch between relative and absolute input. To switch to relative input method in our CursorTap technique, the user places three or more fingers from one hand against the screen, and places a finger of their other hand to create a cursor at the touch point. The movement of that mobile finger controls the movement of the cursor (Figure 3.1 left).

The cursor exists for as long as at least one finger remains on the screen, allowing for a *clutching* movement where the user keeps at least one finger stationary against the screen and repeatedly drags and lifts the finger of the other hand to move the cursor around, much like a smartphone user drags and lifts their finger to scroll down a large document. When all fingers are lifted from the screen, the cursor vanishes and the screen returns to absolute direct input.

To avoid spurious taps caused by the user's palm or other fingers, we create a dead zone encompassing the fingers that are not used to move the cursor. This dead zone is visible to the user, and any finger movement within this zone will not contribute to cursor movement, nor will any taps within this zone cause the cursor to register a tap.

Unlike the devices supported by the pen-based HybridPointing design, most touch screens do not have a method for tracking the position of the user's finger when it is

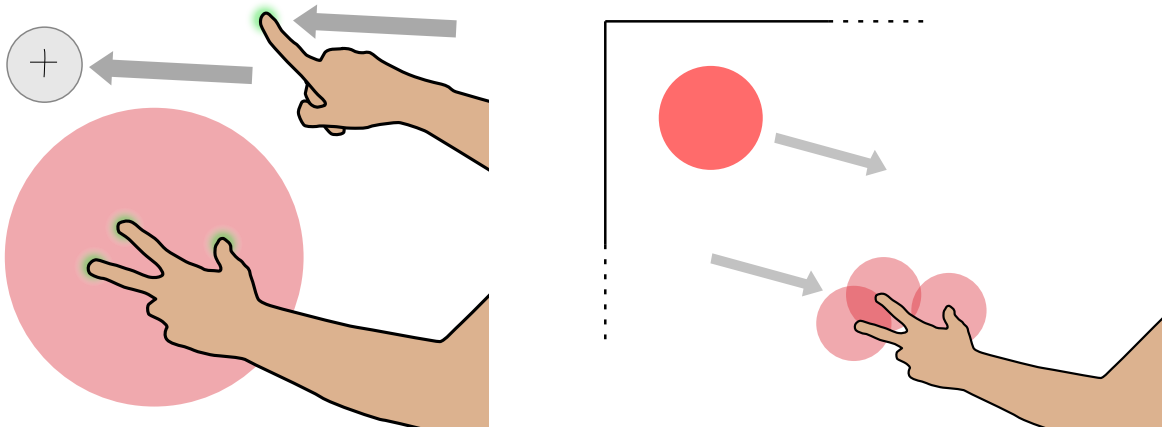

Figure 3.1: Left: CursorTap. The pink area illustrates the dead zone under the user's non-moving fingers, while the movement of the finger outside the dead zone moves the cursor. Right: Drag. The pink areas illustrate the individual dead zones. The red target moves in the direction of the user's fingers, as does a visual representation of the logical border of the screen.

not in contact with the screen. Thus, our design simply allows the user to tap while the cursor is active (i.e. while the stationary fingers are still touching the surface) in order to click at the location of the cursor. This method works well with the use of the off-hand to enter a mode switch, as it means users can simply lift all their fingers to return to absolute input, while still being able to clutch and tap without leaving the relative input mode. We tested another interaction mode where lifting all fingers resulted in a tap at the cursor's location at the same time as mode-switching back to direct input, but we found this to be less useful, as it means the cursor could not be used to select multiple distant targets without having to seek it all the way back to the target each time, and accidental mode-switches always caused a click.

When moving the cursor, we apply a *CD gain* curve, which maps the velocity of the user's finger to the velocity of the cursor. This curve is loosely based on the pointer acceleration used in Windows XP [13], but modified to ensure that the cursor always moves at least as quickly as the user's finger, as that proved to be more intuitive in preliminary studies, as well as adding an upper limit on cursor speed to avoid jumps. The curve used is a sigmoid curve, using the following equation, with v representing the speed of the user's finger:

$$\max\left(L \cdot \text{clamp}\left(\frac{v}{v_1}\right)\left(\frac{1}{1+e^{-k(v-v_0)}}\right)^T, x\right)$$

The clamp function restricts its input to the range $[0, 1]$. The parameters used were $L = 240$, $k = 0.2$, $T = 0.8$, $x_0 = 20$, $x_1 = 4$.

The cursor itself is represented simply as a crosshair. The crosshair is made slightly larger as it moves away from the position where it was originally created, making it easier to see when the user is interacting with distant objects. Furthermore, we draw a circle filled with a semi-translucent colour around the crosshair which also increases in size as the cursor moves further from its initial position. This halo is several times larger than the crosshair, and makes it much easier to find the cursor if the user loses control. The position of the cursor is not constrained to the onscreen area, and a simple triangle is drawn pointing in the direction of the cursor offscreen if it is too close to the border. Unlike the HybridPointing technique, we expect the state of the machine to always be immediately obvious to users; absolute input only occurs when the user has no fingers against the screen, and no cursor is drawn, while relative input occurs only when the user has at least one finger against the screen and a cursor is visible.

Drag

Many smartphones provide an input method for interacting with conceptual surfaces that are larger than the screen itself. By touching and dragging on the screen, the surface is panned (or "scrolled") horizontally and/or vertically to allow users to view and interact with objects that were previously outside the displayed area.

Panning may be an intuitive and common operation, but poses an issue for multi-user environments in large displays; scrolling an object toward one user prevents other users from interacting with that object while it is in use. Further, current implementations of scrolling typically scroll the entire screen at once, which will invariably disrupt any other active users of the screen. Regardless, dragging to pan the entire display is a meaningful baseline for comparison, so we include a Drag technique in our experiment. The user can place two or more fingers on the screen to initiate a drag. Moving the fingers will cause the screen to move along (Figure 3.1 right). Dragging stops when all fingers are lifted.

To reduce fatigue when dragging a 4m wide screen all the way to the user, the movement velocity of the screen is modified using the exact same acceleration function as CursorTap, with the input being the average velocity of the user's fingers. Without this velocity rescaling, the user must repeatedly clutch to drag the far end of the screen all the way to their position.

Like the CursorTap method, we create dead zones around the fingers that are being used to drag the screen. These dead zones are visible to the user, with one dead zone for

each finger, to support dragging with more than one hand. Any touch within any of these dead zones will not register as a tap on a target. A visual representation of the logical border of the screen is drawn, so the user is aware of how far the workspace has been dragged.

3.1 Filtering

As input from the touch driver was fairly noisy and somewhat unreliable, we created a filtering layer. This layer is necessary for the functioning of the multi-touch techniques, as the touch data returned from the driver is especially unreliable in the presence of several nearby touches; in the absence of some form of filtering, spurious finger lifts and movements make it unfeasible to have any mode switch that operates on the number of fingers the user has active. For consistency, this filtering layer was active for all interaction techniques.

See Appendix A for an explanation of how this filtering layer is implemented.

Chapter 4

Experiment 1: Technique Comparison

The goal of our first experiment is to compare the performance of our CursorTap Hybrid-Pointing touch technique to straightforward absolute touch input, as well as Drag, an existing relative input technique. The experiment design is similar to the relative pen input comparison used by Forlines et al. [7].

4.1 Participants

We recruited 24 participants for the study. The participants consisted largely of university students from a broad selection of majors, 10 female and 14 male. Participants were remunerated $15 for what was, on average, one hour of experimentation.

4.2 Apparatus

We used a 413 $\times$ 117 cm wall-sized touch screen made of eight 1920 $\times$ 1080 px screens of approximately 103 $\times$ 58 cm, arranged in four columns and two rows. The screens are bordered by a PQ Labs commercial optical touch frame for detecting input. There are 22 mm wide interior bezels between the tiled displays. We prevented any target from appearing across these bezels, and previous work has shown the effect of bezels is minimal [19]. An optical touch frame of this size introduces spurious touch points and position noise, so we developed our own filtering methods (discussed below).

Following Nancel et al.'s general guidelines [14], the pointer acceleration function was tuned for the display such that the cursor moved approximately 1:1 at low finger speeds, and with higher speed movements the cursor could be placed at any location on the display in a controlled manner without the need for clutching.

4.3 Task

We designed a standard 2D target acquisition task, which required participants to point and click on a series of targets positioned around the screen. The targets were drawn as red circles on a black background. When participants successfully clicked a target, it would turn white and vanish and another target would appear elsewhere on the screen. When the participant missed a target, the event was logged and the target glowed red. Participants had to select each target before continuing in order to prevent them from "racing" through the experiment carelessly. A line to the next target was shown briefly to make it easier to find the target when it appeared. If the DRAG technique caused a target to no longer be visible on-screen, a red arrow pointing to the off-screen position of the target was drawn.

In order to maximize the use of our screen space, we required participants to stand at one side of the screen when using relative input techniques. The standing zone was marked by two parallel lines 1.03m apart on the floor (Figure 4.1). Participants were told to keep their feet within the assigned zone, but could stretch their body outside if they wished. This simulates a larger touch screen, where a user stands at a primary location to perform local manipulation, but uses relative input to select targets far away.

The targets themselves were arranged at the corners of a rectangle, starting from the upper-left and moving down, up, down, right, left, right, up, down, up, left, right, left, arriving back at the starting position. Participants were informed that the targets would follow this pattern. The right and left movements of the target were reversed when the participant began on the right side of the screen, with the target beginning in the upper-right corner, such that the first four targets were always in front of the user and within reach. The corners of the rectangle that did not currently house the active target were drawn using a faint grey, so that participants would be able to quickly anticipate the next position of the target. The motivation behind this design is to reduce the amount of time spent by participants attempting to find the next target, which would impact the measurement of pure targeting performance. Thus, our design differs from that of Forlines et al., which did not give an indication of where new targets would appear, and positioned targets randomly. Furthermore, in our experiments, only purely horizontal

and vertical target movements were tested, as the aspect ratio of the apparatus was very wide, so that near 45° diagonal movements would still be achievable without resorting to distance interaction techniques.

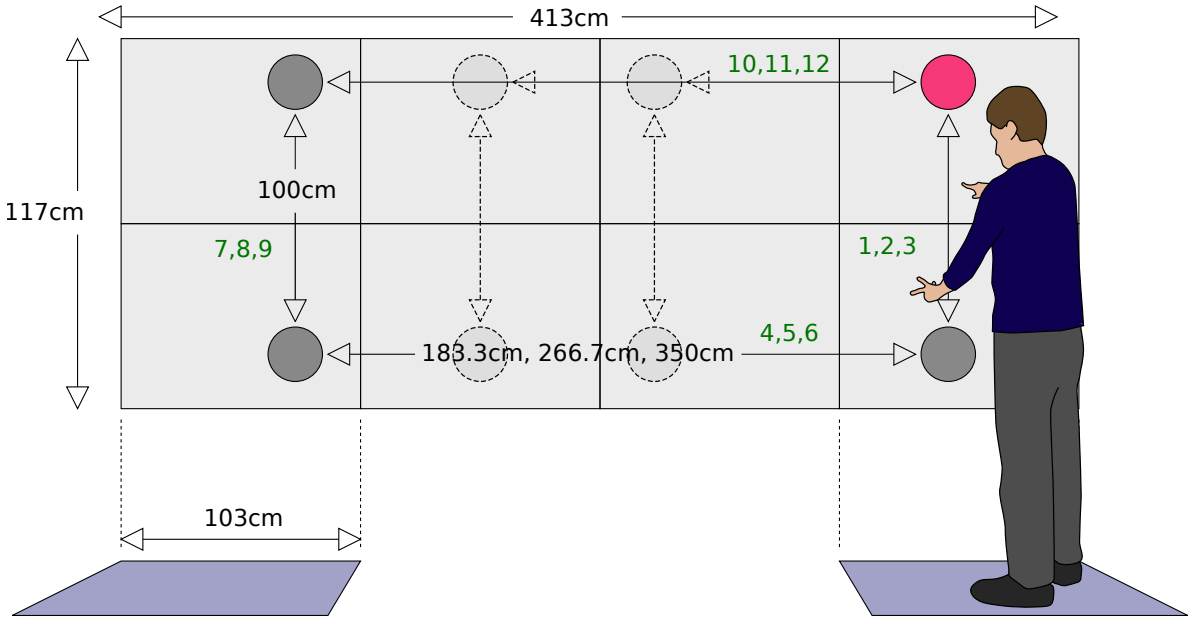

Figure 4.1: Task setup showing target positions for the three long distances and selection pattern starting from upper left; the lower two light-blue squares represent the side standing zones.

While Forlines et al. compared the performance of the hybrid method against both absolute and relative input methods, the use of our cursor method in a purely-relative trial is not particularly relevant, as the technique is designed to be used alongside absolute input in ordinary use. Instead, we compared three input METHODS: ABSOLUTE, where participants would directly tap targets to select them and were not required to stand inside one of the zones; CURSORTAP, where participants used the hybrid input method with mode switching; and DRAG, the other baseline technique described above.

4.4 Design

We used a repeated-measures design with within-subject factors, as opposed to the between-subject factor of input mapping used in the HybridPointing study. Our

independent variables were METHOD (ABSOLUTE, DRAG, CURSORTAP), target WIDTH (5.1 cm, 10.2 cm) and target DISTANCE (100 cm, 183.3 cm, 266.7 cm, 350 cm). The combination of width and distance creates IDs ranging from 3.4 to 5.9.

Forlines et al. had a target distance of 400 cm, but this was larger than our display, so we distributed target distances equally from 100cm to 350cm. Note that the target distance of 100 cm was not used on its own, but rather was used in every rectangle as the vertical distance between targets. The other three target distances were used as the horizontal distances between targets.

Each participant performed 6 blocks of target selections with each METHOD. 12 timed trials were made for each combination of target WIDTH and DISTANCE, aside from the shortest target distance.

Half of the participants started on the right side of the display by standing in the corresponding zone (see Figure 4.1), the other half on the left. When standing in the left zone, the left side of the rectangle always coincided with the middle of the leftmost column of monitors. The right side corresponded to one of the three distances for the remote targets. The conditions were mirrored when participants were standing on the right side. People switched sides for each block. Combined with the six possible permutations of the order of *input method*, this resulted in twelve total conditions across the 24 participants. Participants were permitted to take breaks in between each block.

The target widths and target distances used within a block were arranged in a random ordering which was kept consistent between participants, such that the first block of each trial always used the same ordering of widths and distances, regardless of which interaction method was used first.

In summary: 3 METHODS $\times$ 6 blocks $\times$ 2 WIDTHS $\times$ (3 long DISTANCES $\times$ 6 selection + 18 short DISTANCE selections) = 1296 selections per participant.

Source code for the trial program is available.[1]

[1] Source code will be made available on github.

Chapter 5

Results

Repeated measures ANOVA were conducted for all measures[1] and when significant effects were discovered, post hoc pairwise t-tests with Holm correction were performed. Because measures for preference exhibited non-normality, they were transformed using Aligned Rank Transform [20]. Trials were aggregated by participant and the factors being analysed. Time data were aggregated using the median to account for skewed distribution.

For each participant, target times more than 3 standard deviations from the mean of the participant's time were excluded as outliers. In total, 405 trials (1.3%) were removed.

5.1 Learning Effect

Overall, BLOCK had a main effect on selection time ($F_{5,115} = 9.41$, $p < .0001$, $\eta_p^2 = .045$), and on error rate ($F_{5,115} = 2.92$, $p = .015$, $\eta_p^2 = .048$). For time, post hoc tests found blocks 1, 2 were significantly slower than blocks 3 to 6 (all $p < .02$), suggesting an initial learning effect. For error, the only difference was between block 2 and 5, but combined block error rates are all below 4%. In subsequent analysis, we use only blocks 3 to 6 as more representative of practised performance. This is similar to Forlines et al., who removed the first 4 blocks to account for learning.

[1] When the assumption of sphericity was violated, we corrected the degrees of freedom using Greenhouse-Geisser (Greenhouse-Geisser's $\epsilon < 0.75$) or Huynh-Feldt (Greenhouse-Geisser's $\epsilon \geq 0.75$).

5.2 Error Rate

If a target was missed one or more times, the trial was marked as an error. High velocity movements occasionally registered false taps as errors without affecting user behaviour in that trial. We corrected for 1649 of these by identifying taps more than one target diameter away from the outer edge of the target. The error rate is the ratio of error trials to all trials.

There was a main effect of TECHNIQUE on error rate ($F_{2,46} = 31.45$, $p < .0001$, $\eta_p^2 = 0.41$). Post hoc tests show the error rate for DRAG (0.5%) is lower than both ABSOLUTE (4.7%) and CURSORTAP (5.4%) (all $p < .0001$). The main effect provides a high-level summary, but the effect of technique on width is perhaps more relevant. There was an interaction between TECHNIQUE and WIDTH on error rate ($F_{2,46} = 19.83$, $p = .0001$, $\eta_p^2 = 0.12$) (see Figure 5.1). As expected, error rates increased with smaller widths for CURSORTAP and ABSOLUTE (all $p < .0001$, but there was no significant difference across widths for DRAG. There was no significant difference between CURSORTAP and ABSOLUTE at either width.

These are all very good error rates, but the rate for DRAG is remarkably low. We believe the benefit of always bringing the target directly in front of the body is the reason. Note that Forlines et al. observed overall error rates between 4.2% and 6.8%, but no breakdown by width is reported.

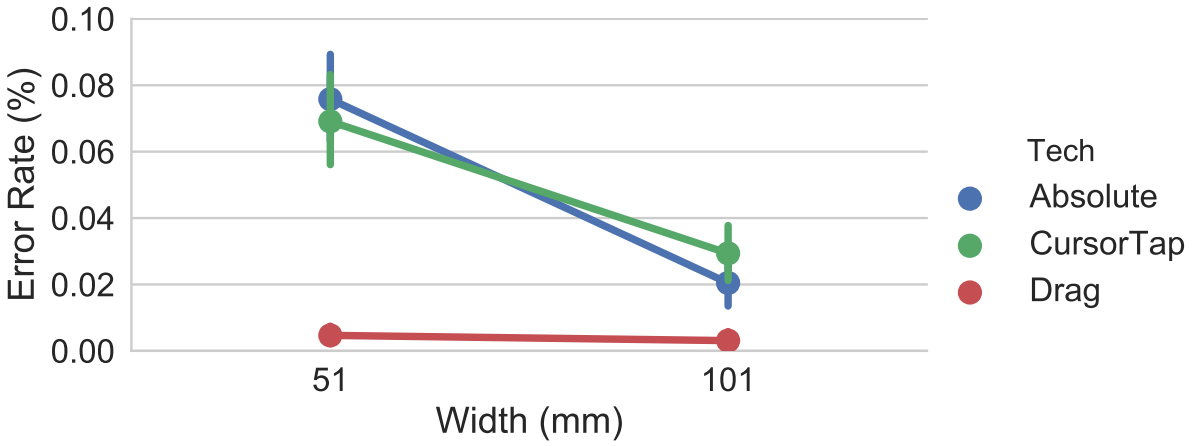

Figure 5.1: Error rate for TECHNIQUE by WIDTH.

5.3 Selection Time

Overall, when all tested conditions are combined, TECHNIQUE had a significant main effect on time ($F_{2,46} = 24.45$, $p = .0001$, $\eta_p^2 = 0.31$) with post hoc tests showing CURSORTAP to be 500 ms faster than ABSOLUTE and DRAG (both $p < .0001$). However, the significant interaction of TECHNIQUE and DISTANCE is most relevant ($F_{3.78,87.03} = 73.99$, $p = .0001$, $\eta_p^2 = 0.46$) (see Figure 5.2). At the shortest DISTANCE of 1000mm, the time for ABSOLUTE (1011ms) is significantly faster than both CURSORTAP (1638ms) and DRAG (1624ms) (all $p < .0001$). But at a DISTANCE of 1833mm, times for ABSOLUTE and CURSORTAP are no longer significantly different (both near 2200ms) and DRAG is slightly, but significantly, slower (2488ms) (all $p < .01$). At the second longest DISTANCE of 2667mm, the time for CURSORTAP (2169ms) becomes significantly faster than both ABSOLUTE (3082ms) and DRAG (2695ms) (all $p < .0001$). Finally, the time for CURSORTAP (1777ms) diverges even more at the longest distance of 3500mm, becoming approximately half of the times for ABSOLUTE (3797ms) and DRAG (3253ms).

The results suggest CURSORTAP times remain nearly constant as the distance increases, but both ABSOLUTE and DRAG increase with distance. An approximately linear increase for ABSOLUTE was also noted in Forlines et al., but the nearly flat response of CURSORTAP shows a much stronger benefit for touch hybrid pointing than pen hybrid pointing, which still showed a linear increase.

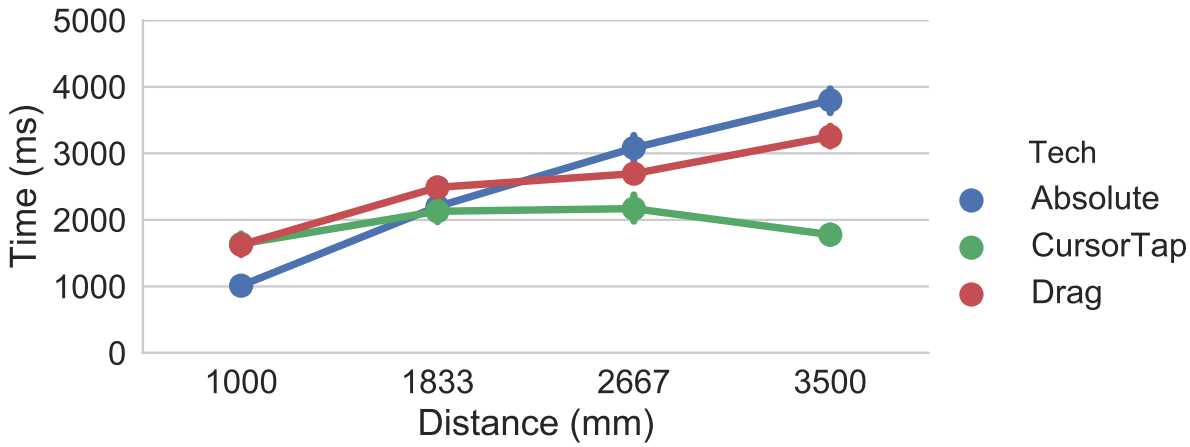

Figure 5.2: Target time for TECHNIQUE by DISTANCE.

5.4 Hybrid Technique Analysis

We analysed users' mode switching behaviour between absolute and relative input modes during the CursorTap condition (see Figure 5.3). This measures the distance between the participant and the target selected, where participant position is estimated as the centre of the bounding box of all the user's active touches when the most recent cursor was spawned. A mode switch of AllAbsolute or AllRelative indicates the participant only had absolute or relative input active during the selection of the target, while ToRelative and ToAbsolute indicate that the user created and used a cursor, or stopped using an active cursor and touched the target directly, respectively. Multiple indicates that the participant switched modes more than once during the target selection.

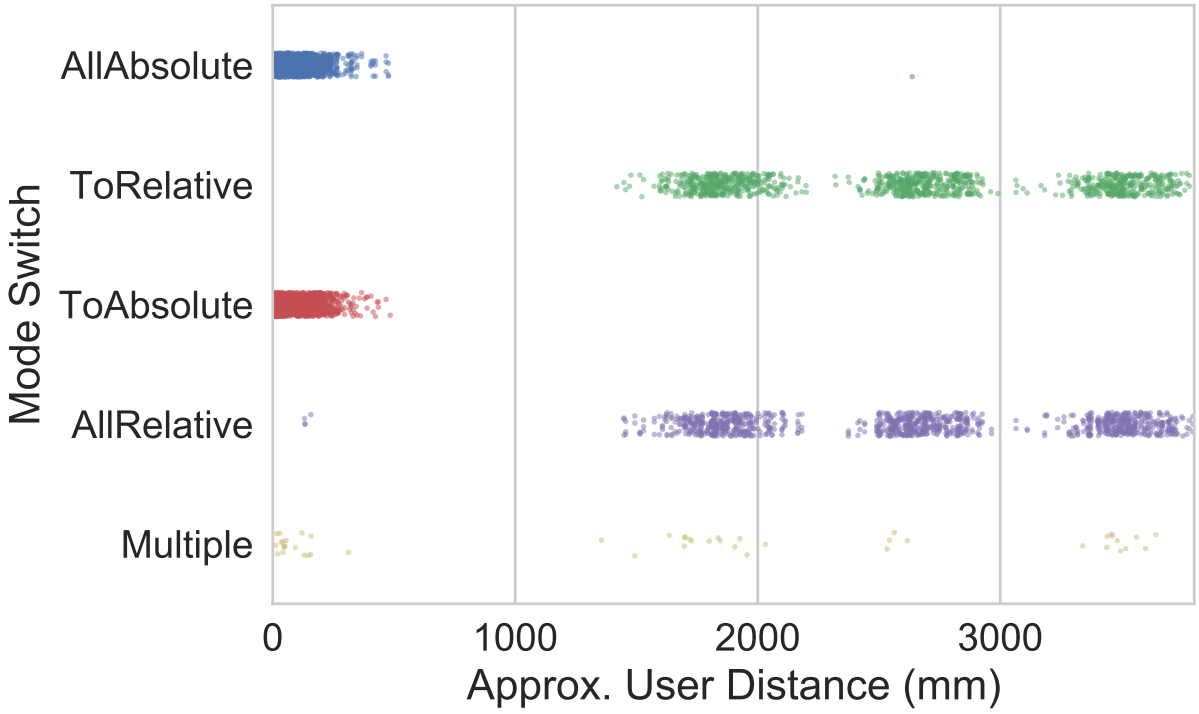

Figure 5.3: CursorTap mode switches by distance from user to target.

Figure 5.4 shows the time taken to select a target, by the combination of mode and the distance between targets. [2] [3] A target which appears in front of the user is universally

[2] Absolute-Far is not shown, as participants could not use absolute input to select distant targets.
[3] Relative-Near is not shown, as only five datapoints were present for participants using the relative

much faster to select than a target which appears far away from the user; in particular, even when the distance between targets is only 100 cm (i.e., vertical movement between targets), it is much faster for the user to tap directly on the targets than to move the cursor from one target to the next.

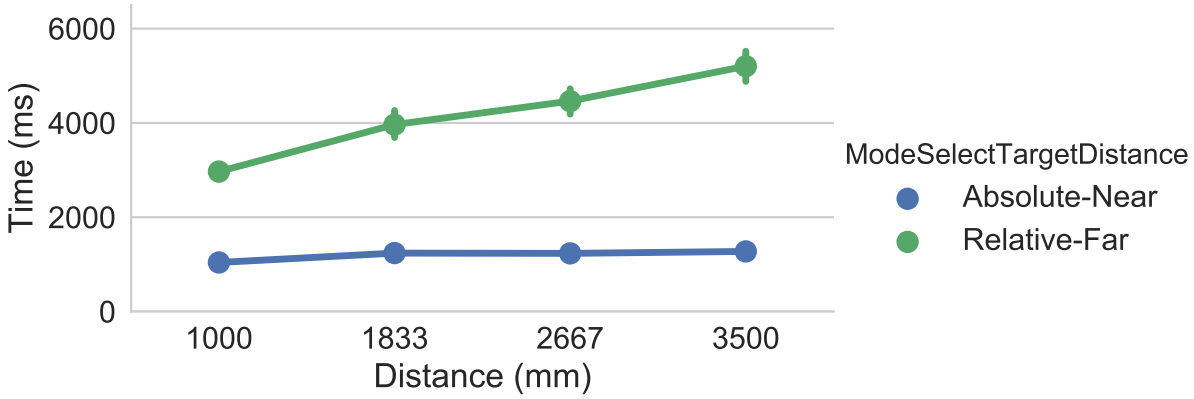

Figure 5.4: CursorTap time by distance for combinations of selection mode and distance from user to target.

Figure 5.5 shows the error rate for selections by their mode switching behaviour. Figure 5.6 breaks this down by the distance between consecutive targets. AllAbsolute only contains samples for the shortest distance, as participants using absolute input to select two subsequent targets necessarily involved both targets being near the participant; while participants could feasibly have used absolute input to select a target 183.3 cm from their standing position, participants generally chose not to. Likewise, AllRelative contains very few data points for long distances, as participants rarely chose to use relative cursor input to select two targets with a large distance between them, instead opting to select the nearby target with absolute input; ToRelative contains no data points for the shortest distance, as participants never used relative input to select a nearby target after the previous target was also nearby, and when two targets had the shortest distance between them but were far from the participant, they will have used relative input to select the previous target, making it necessarily an AllRelative or Multiple selection.

Figure 5.7 shows a Fitts' Law regression, taking a linear regression of the time taken to select a target against its *Index of Difficulty*. [6] For a more controlled regression, we examine only trials which had a ToRelative mode switch, where participants began

cursor mode to select a target immediately in front.

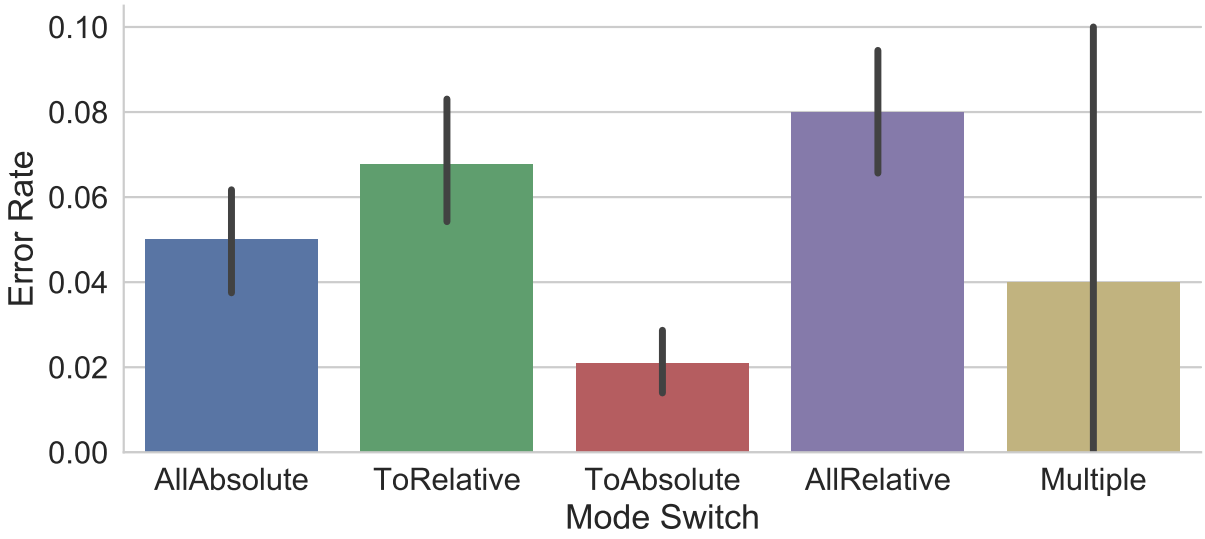

Figure 5.5: CursorTap error rate by mode switch type.

without a cursor and created one to select the target. We compute the Index of Difficulty using the distance from the cursor's spawn position to the target, and only begin measuring the time taken to select the target when the cursor is spawned.

Regression gives a slope of 293 ms, and an intercept of 1844 ms ($r = 0.24$, $p < 0.0001$, $SE = 34.42$).

5.5 Preference

At the end of the experiment, participants were asked to consider accuracy, speed, and fatigue and provide a numerical preference score for each technique. Preference scores used a real numbered, continuous scale from 1 (most preferred) to 7 (least preferred). Decimal ratings such as 4.5 were permitted.

Differences between techniques across all metrics were significant (*all* $p < 0.005$). Post-hoc tests showed a significant preference for Absolute and Drag over CursorTap in fatigue and accuracy, with no significant preference between Absolute and Drag; a significant preference for Drag over CursorTap and Absolute with no significant preference between CursorTap and Absolute; And a significant preference in mean rating for Drag over both CursorTap and Absolute, with no significant difference in mean rating between CursorTap and Absolute.

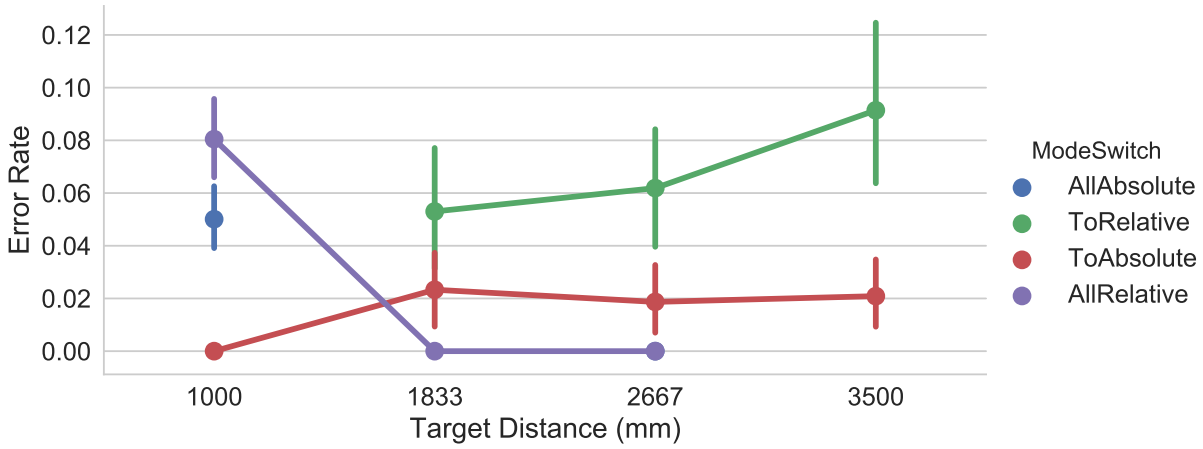

Figure 5.6: CursorTap error rate by target distance, for each mode switch type. Multiple switches excluded due to high error rates.

	Fatigue	Speed	Accuracy	Mean
Absolute	4.6±0.36	4.0±0.36	2.3±0.37	3.6±0.24
Drag	2.3±0.21	2.7±0.34	2.6±0.30	2.5±0.16
CursorTap	3.5±0.38	4.3±0.30	4.7±0.32	4.2±0.20
Mean	3.2±0.22	3.5±0.21	3.7±0.23	3.5±0.12

Table 5.1: Average rating provided by participants, along with standard error of the mean, for the given subjective factors.

5.5.1 Experiment Discussion

While participants were generally not favourable toward CursorTap relative to the two other interaction techniques, performance data demonstrates that CursorTap is not much slower than either of the other techniques at short distances, and overtakes them greatly as distance increases. Likewise, error rates for CursorTap are not much higher than the other two techniques for the common scenario of large targets on a large screen. We hypothesize the gap will close further as large touchscreens become less error-prone, as we observed more points of failure for the CursorTap technique relative to Drag or Absolute in the face of poor data from the input driver. In particular, large spurious jumps in input fed into the cursor acceleration function and became larger, which was then mitigated by using the average movement across fingers for the Drag technique.

CursorTap's error rates demonstrate interesting behaviour when broken down by mode switching behaviour. In particular, while a significant effect was found of Mode Switch on

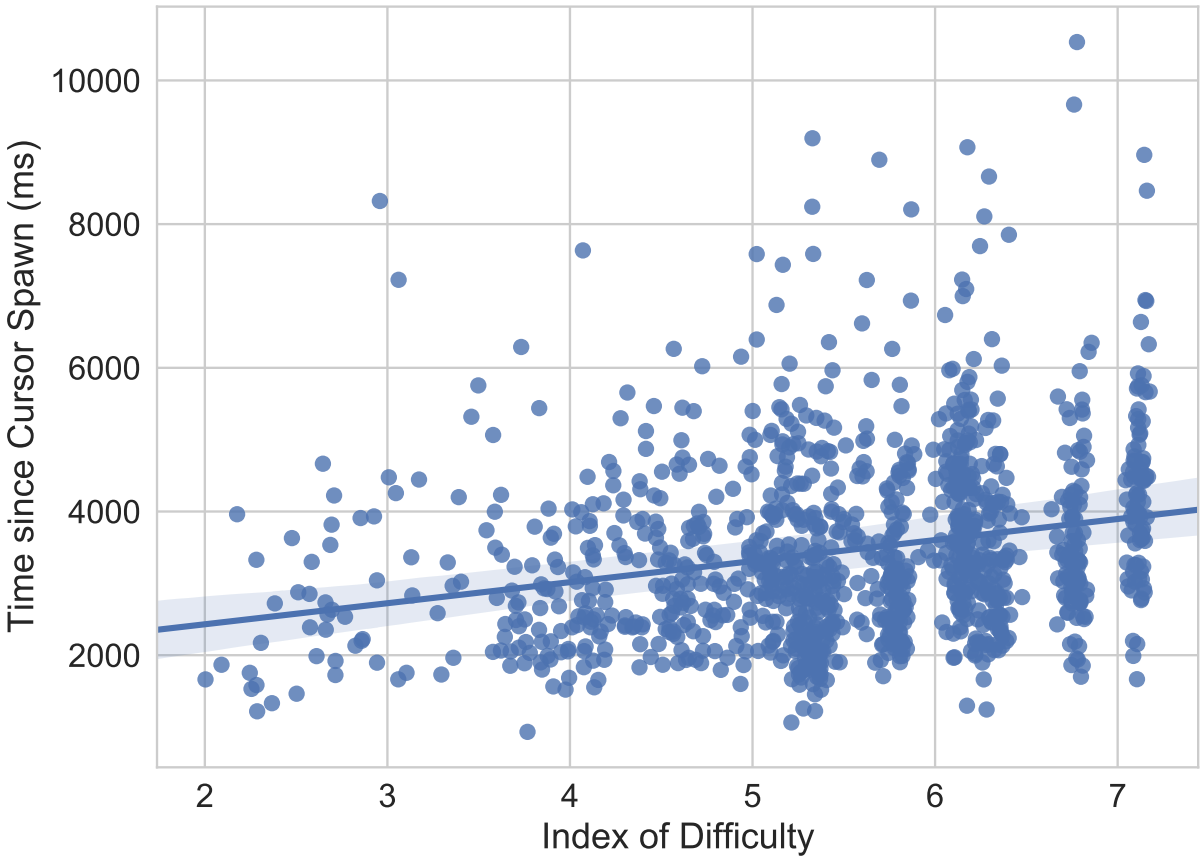

Figure 5.7: Fitts' law regression of ToRelative **trials.**

Error ($F_{3,69} = 12.43$, $p < .0001$, $\eta_p^2 = .217$)[4], post-hoc analysis shows no significant difference between All Absolute and To Relative selections (p = 0.58). This may suggest that participants find it more intuitive to send a cursor out and select a distant target than trying to move an already distant cursor to select a target near their previous target. This supports our hypothesis that the CursorTap technique performs best for the common scenario of selecting a distant target, followed by performing absolute selections on nearby targets. This suggests some interesting user interface designs, which we explore in the discussion section.

Participants experienced very little error with the Drag technique, and its speed was higher than the Absolute technique for the two larger distances. This suggests that the

[4]Multiple excluded from graph due to few datapoints and participants without data.

Drag technique could be useful in a single-user setting; however, this technique becomes problematic in a multi-user setting due to the fact that multiple users may need to move the screen to different locations to interact with targets, or depending on the application, may find it useful to interact with the same target simultaneously.

This highly controlled experiment shows a benefit for CursorTap, but we wondered how (and if) people would use it in a real setting, in particular when sharing the display. Our second study addresses this open question.

Chapter 6

Experiment 2: Usage in a Realistic Setting

The goal of the second experiment is to explore how users engage with the *CursorTap* technique in a more realistic situation, including sharing the large display with another person. We use a game setting to encourage different types of technique usage and to control cooperative and competitive behaviour between the two users.

6.1 Participants

We recruited 14 participants for the study. The participants consisted largely of university students, from a broad selection of majors. Participants were remunerated $10 for what was, on average, one hour of experimentation. Participants completed the experiment as pairs. Those pairs consisted of strangers, acquaintances, friends, and couples.

6.2 Apparatus

The same apparatus and interaction technique code as experiment 1 was used.

6.3 Game Task

To test natural usage of the technique, we created a simple invasion game for two participants to play simultaneously. The game is very simple, consisting of three different colours of enemies (represented by small circles) appearing past the edge of the screen on either side and slowly moving toward the planet Earth in the centre of the screen. Enemies could be eliminated by the participants, preventing them from colliding with Earth. The Earth acquires a brown tint as more enemies collide with it. Participants would lose points when enemies collided with Earth, and gain points for eliminating them. Two conditions were tested: one where participants' scores were summed and displayed as a single shared score, and one where participants' scores were displayed separately. We refer to those conditions as COOPERATIVE and COMPETITIVE, respectively.

Each participant had a *workspace*, a visually delineated area on the touchscreen. Participants were informed that they were not permitted to touch inside the other participant's workspace; therefore, a touch initiated within a participant's workspace was coded as being caused by that participant's hand. The workspace covers the entire vertical height of the screen, and extends from one side of the screen to near the centre line.

The gap between workspaces in the centre of the screen is a neutral zone. Within this zone are three weapons, represented by distinct circular designs with unique colours. Each weapon can be used to eliminate enemies in an area not covered by either participant's workspace. There were three weapons with distinct colours. The weapons were to be used to eliminate aliens matching their colour. Players could select a weapon by tapping it with a finger and dragging it into their own workspace, or by using the *CursorTap* technique inside their workspace to spawn a cursor and clicking to acquire the weapon. This action assigned the weapon to the participant, thereby also preventing the other player from using it.

The weapon could be moved by directly tapping and dragging it around with a finger, or again with *CursorTap*, with the weapon underneath the participant's cursor. The three weapons were used differently to test different aspects of direct manipulation:

- The green weapon was called the SHIELD, and any green alien which collided with the shield was eliminated, as long as the shield was being used by the participant. This tested the participant's ability to *move* their interaction to a specific area.
- The red weapon was called the CANNON, and when it was tapped, either by the user's finger or by tapping with relative input technique active, any red enemies which

were beneath the tool were eliminated. This tested the participant's ability to *target and click*.

- The blue weapon was called the BLACK HOLE, and displayed a trail behind it when dragged around the screen. When this trail intersected itself, any blue enemies within the loop were eliminated. This tested the participant's ability to perform a *lasso-style selection*.

These different tools tested common interaction modes which may be useful in real-world situations, such as multi-user image editing. Additionally, all of those tools functioned identically whether they were being used with absolute input or relative input, allowing a reasonable comparison between the two techniques.

To add interest to the game and facilitate tool switching, a fourth tool, the MAGNET, was added. The magnet was not used to defeat any enemies, and no enemies were assigned its colour. Instead, by selecting the magnet tool and placing it in the other participant's workspace, the player can steal the weapon, causing the magnet to return to the centre of the screen. As with the other weapons, moving the magnet could be done with absolute tap and drag or with the relative mode. Dropping the magnet into another participant's workspace is the only time when a participant's finger may enter the other user's workspace.

Enemies appeared in waves, with one wave coming from each side of the screen. A wave consisted of a large number of enemies of a primary type, a moderate number of enemies of a secondary type, and a small number of enemies of a third type. The primary type of the wave on the left side of the screen was the secondary type of the wave on the right side of the screen, and vice-versa. The third type was the same on both sides of the screen. This design was meant to encourage participants to change tools and to see what would happen if both participants needed to switch tools simultaneously. For a given side, 12 enemies of the primary type, 4 enemies of the secondary type and 2 enemies of the tertiary type were spawned in a wave, for 18 targets total on a side.

The goal of the game was to clear all enemies. A wave ended when all the enemies in the wave had been eliminated on both sides either by being defeated with a weapon or by colliding with the Earth. The next wave began immediately after. There was no victory or loss condition; additionally, there was no lower bound on how low scores could go.

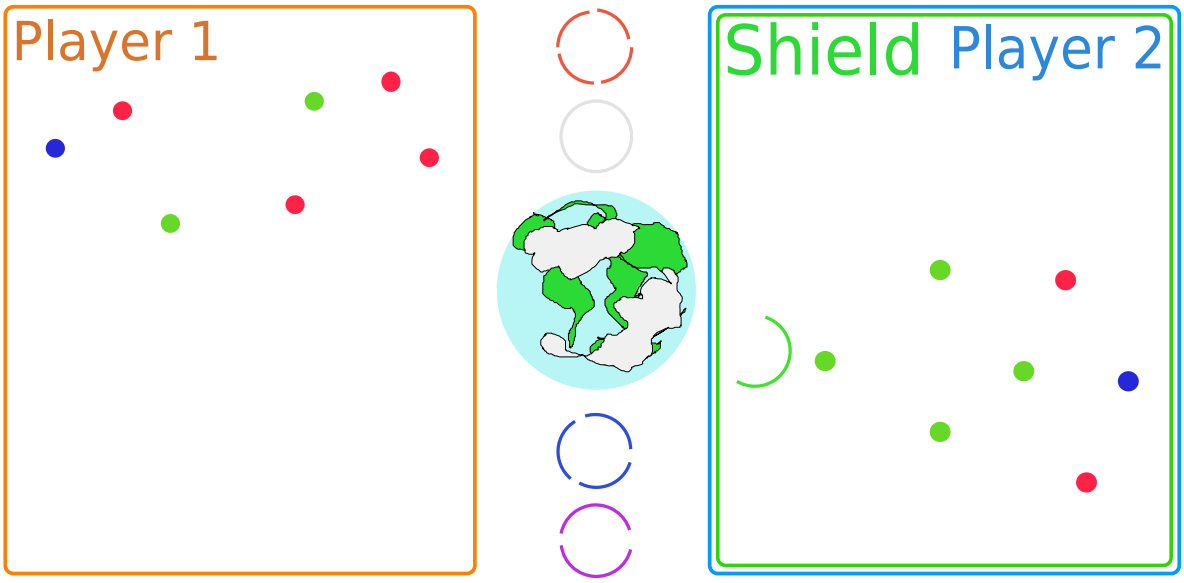

Figure 6.1: A depiction of the game, with the size of interface elements exaggerated for visibility. Player 2 has the Shield tool active. By moving it with their finger, or activating the relative input method, Player 2 can eliminate the green enemies. Player 2 cannot move their fingers into Player 1's workspace. If the enemies aren't eliminated, they will slowly move inward and collide with Earth.

6.3.1 Design

The game was run twice for each pair, once in the cooperative condition and once in the competitive condition. Within each run of the game, we had two BLOCKS of six WAVES, giving the participants an opportunity to rest between blocks.

In summary:

2 CONDITIONS × 2 blocks × 6 waves × 2 sides × 18 targets = 864 potential selections per pair, across 7 pairs.

Source code for the game is available.[1]

[1] Source code will be made available on github.

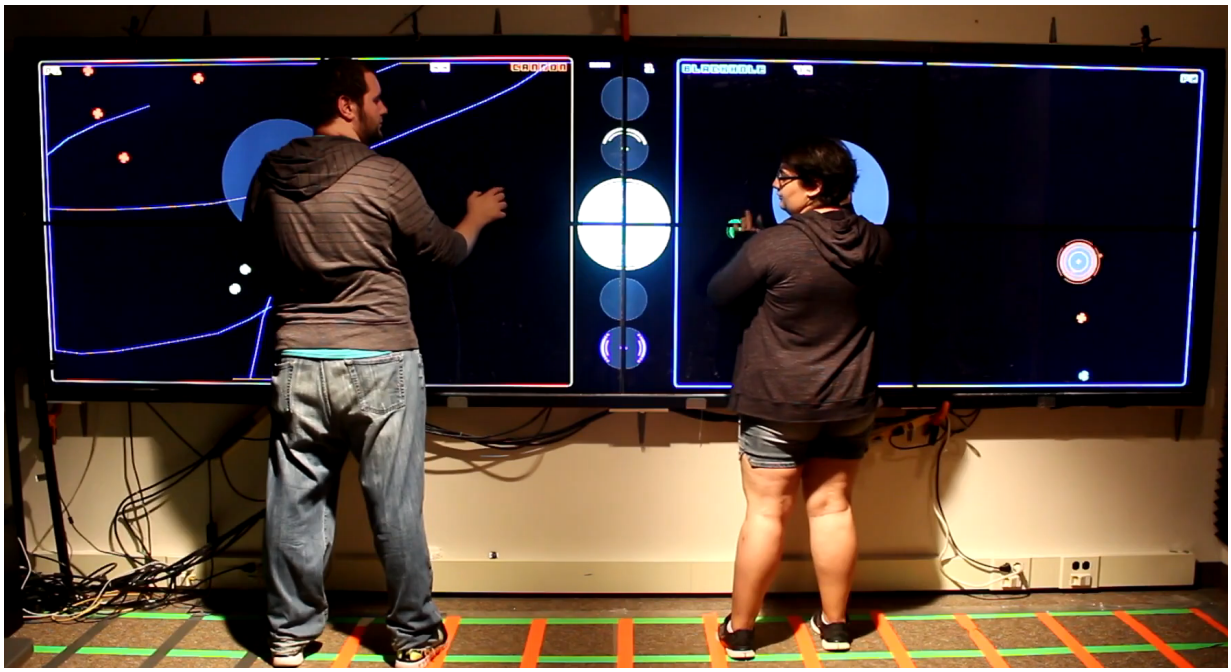

Figure 6.2: A photo of the game being played.

Chapter 7

Experiment 2 Results

Repeated measures ANOVA were conducted for all measures[1] and when significant effects were discovered, post hoc pairwise t-tests with Holm correction were performed. Target selections were aggregated by participant and the factors being analysed. Time data were aggregated using the median to account for skewed distribution. Because measures for preference exhibited non-normality, they were transformed using Aligned Rank Transform [20].

7.1 Learning Effect

No learning effect was observed within participants between the first block and the second block in the median amount of time taken to eliminate a target since its spawn ($p > 0.05$).

7.2 Use of the Hybrid Technique

Across all BLOCK and WEAPON, 8 of the 14 participants used the hybrid technique to eliminate enemies at least 5% of the time. The remaining analysis will be across these 8 participants.

Participants regularly used the hybrid technique to eliminate enemies using relative mode instead of using the absolute mode, with every tool. Of these 8 participants, the

[1] When the assumption of sphericity was violated, we corrected the degrees of freedom using Greenhouse-Geisser (Greenhouse-Geisser's $\epsilon < 0.75$) or Huynh-Feldt (Greenhouse-Geisser's $\epsilon \geq 0.75$).

relative mode was used to eliminate 51.5% of enemy targets, with 41.5% of CANNON enemies, 50.6% of SHIELD enemies, and 62.6% of BLACK HOLE enemies. There was a significant effect of ENEMY TYPE on MODE ($F_{2,14} = 4.80$, $p < 0.026$, $\eta_p^2 = 0.114$), with post-hoc tests showing that participants preferred using the relative mode technique for BLACK HOLE over SHIELD over CANNON (all comparisons $p < 0.001$).

7.2.1 Effect of Distance

61.4% of the time, the relative mode was used to eliminate an enemy target which was on the opposite side of the screen (and therefore, the participant could not have used direct tap to eliminate that enemy, as they were constrained to their own workspace.) The other 38.6% of the time, the relative mode technique was still used, despite the enemy being within the participant's workspace. This use of the relative mode to eliminate an enemy within the participant's own workspace accounted for 19.8% of all enemy eliminations (15.6%, 17.2%, 27.1% for the CANNON, SHIELD and BLACK HOLE enemies respectively.)

There was no significant effect of BLOCK on MODE. There was no significant effect of MODE on ENEMY LIVE TIME.

7.3 Specific Tools

Participants preferred to use the BLACK HOLE tool using the relative mode more than the others, even when the targets being encircled were within the participant's own workspace, as seen above, where the tool appeared more frequently than the others in every metric. A simple hypothesis for this is that manipulating the cursor allows users to encircle a much larger area with smaller hand movements, allowing users to select more of the black hole enemies simultaneously than they would by encircling them with absolute touch input. However, observing targets killed in a single black hole loop shows no significant effect of MODE on TARGETS KILLED (examining only those participants who used both the absolute mode and the relative mode, [2] $F_{1,6} = 0.143$, $p = 0.71$, $\eta_p^2 = 0.011$).

Participants used the CANNON tool in relative mode less frequently than all other tools for selecting targets. This may due to the fact that the cannon tool was the only tool of the three used to select targets that required a tap. In particular, as there was no penalty

[2] One participant only ever used the relative mode to interact with the black hole tool.

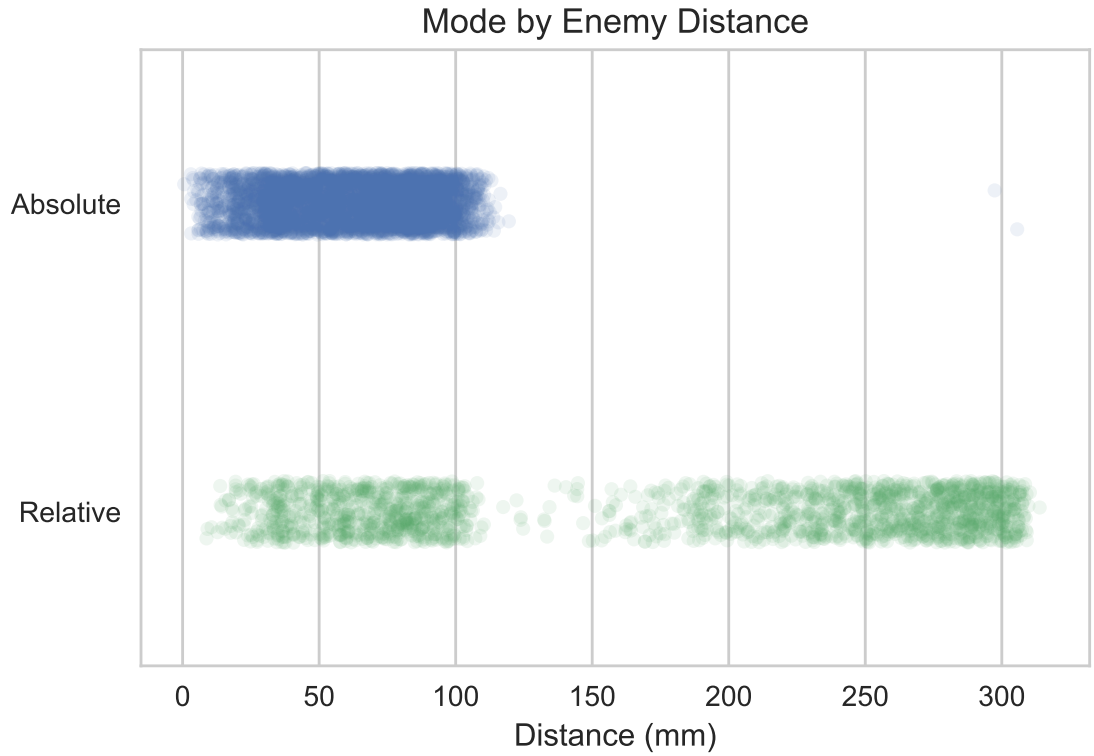

Figure 7.1: Density plot of enemy distance from centre of workspace at time of defeat, separated by whether the relative mode was used or not.

for false negatives (a tap that defeated no enemies), participants may have found the dead zone created around the user's non-active fingers to be a nuisance for tapping, instead of helpful. This may have discouraged them from using relative input for tap-based interaction, as no dead zones were active during the absolute mode.

7.4 Cooperative vs. Competitive Condition

CONDITION had no significant effect on MODE ($F_{1,7} = 0.733$, $p = 0.42$, $\eta_p^2 = 0.0196$), nor did it have an effect on TARGET ON SAME SIDE ($F_{1,7} = 0.00018$, $p = 0.99$, $\eta_p^2 = 0.0000$) within targets that were selected using the relative mode. Additionally, examining whether a defeated enemy was in the most frequent to least frequent enemy type within a given wave on a given side of the screen, MODE had no effect on TARGET TYPE FREQUENCY ($F_{1,7} = 0.019$,

$p = 0.893$, $\eta_p^2 = 0.0016$). From observation, participants did not seem to demonstrate significantly different behaviours based on the condition. This is surprising, as one might have expected participants to optimize their behaviour for a higher combined score in the COOPERATIVE condition.

7.5 Difficulty

Participants generally did not find it difficult to eliminate all the targets within the allotted time. In only 3 of the 7 trials, the number of targets selected was less than the total of 864 targets, with the minimum number of targets selected being 852. CONDITION had no significant effect on the number of targets that were successfully selected ($F_{1,6} = 0.095$, $p = 0.767$, $\eta_p^2 = 0.0017$).

7.6 Preference

At the end of the experiment, participants were asked to consider accuracy, speed, and fatigue and provide a numerical preference score for each combination of weapon and use of absolute vs. relative input. Preference scores used a real numbered, continuous scale from 1 (most preferred) to 7 (least preferred). Decimal ratings such as 4.5 were permitted. (see Table 7.1).

We found that users did not have a particularly strong preference for either input type, with no individual metric having a statistically significant difference between them. Similarly, there was not statistically significant differences when grouped by techniques, metrics, or both.

Although not significantly different, our data reveal a trend that the relative mode is considered less fatiguing and faster, but less accurate, than the absolute mode, in almost every cell. This is consistent with conclusions drawn from the quantitative data in Experiment 1. The BLACK HOLE tool is the only tool that defeats enemies whose relative mode rating is at least as good as absolute across every metric, providing mild support to the hypothesis that players used it in relative mode more frequently than other tools to select targets because they found it easier to use in that way. Likewise, the CANNON tool is the only tool whose mean rating is worse overall for the relative mode than the absolute mode, which may go some way toward explaining why it was used least frequently in relative mode.

	Fatigue	Speed	Accuracy	Mean
C, A	3.4±0.51	4.2±0.43	3.2±0.65	3.6±0.33
C, R	4.1±0.53	3.5±0.61	4.2±0.55	3.7±0.32
S, A	3.6±0.60	3.9±0.50	3.3±0.63	3.5±0.32
S, R	3.0±0.44	2.9±0.54	4.0±0.51	3.1±0.29
B, A	3.2±0.43	3.4±0.54	3.5±0.53	3.3±0.28
B, R	2.7±0.51	2.9±0.51	3.5±0.59	3.0±0.30
M, A	2.8±0.55	3.0±0.53	3.1±0.62	2.9±0.32
M, R	2.7±0.53	2.9±0.54	2.9±0.57	2.8±0.31
-, A	3.2±0.26	3.6±0.25	3.2±0.29	3.3±0.15
-, R	3.0±0.25	3.1±0.27	3.6±0.27	3.2±0.15
-, -	3.1±0.18	3.3±0.18	3.4±0.20	3.3±0.11

Table 7.1: Average rating provided by participants, along with standard error of the mean, for the given subjective factors. C, S, B and M refer to the weapon being judged including the magnet tool, while A and R refer to absolute and relative mode. - refers to the mean across measures.

7.7 Experiment Discussion

Our data support the hypothesis that users are willing to use the hybrid cursor technique to interact with all of the interaction modes demonstrated, and that among those who use the technique, they use it regularly. Interestingly, users also used the cursor technique to eliminate targets that were within their workspace, and should have been within reach with minimal movement. Observationally, users took a bit of time to become comfortable with the cursor itself, but very quickly learned to perform the mode switch gesture that creates the cursor.

No significant difference was observed between cooperative and competitive settings, suggesting that the CursorTap technique is suitable for use both in collaborative tasks such as multi-user document editing, as well as game-like competitive activities, or contexts with multiple users performing separate individual tasks on a shared display such as finding information on a mall kiosk.

There were a few participants who did not use the relative input mode at least 5% of the time for every WEAPON and BLOCK. From user feedback, this was usually due to frustration with noisy input from the touch driver, which made the relative input mode feel less stable than absolute input.

Chapter 8

Discussion

Direct tapping and relative cursor control are two techniques that each shine in their respective targeting contexts: Direct tap for close targets and relative pointing for remote targets. CursorTap does not perform well when only a single category of targets is present, especially when those targets are at a critical distance, where the user has to decide which input method to choose. CursorTap performs best when selections involve combinations of far and near targets where the decision to directly tap or use a relative cursor is always clear. In our game, the two exclusive and clearly separated player zones facilitated that decision. We surmise that CursorTap would prove useful in similar multi-user scenarios with personal spaces directly in front of the owner which can be manipulated using classic direct touch input, and with the possibility to occasionally reach to other users' spaces to share content or access a specific tool (similar to our magnet to steal the other player's weapon).

8.1 Possible Improvements

Variations on the hybrid technique other than those tested were considered, based on observation of study participants as well as feedback from the participants themselves.

One variation that was not implemented due to time constraints is a cursor technique where the cursor's position is saved while the cursor is not in use, rather than always being spawned at the location of the user's instigating finger. This can potentially be useful in the case where the user is going to be interacting with many things in the same, distant, area. It would potentially have improved participants' speed in the quantitative

study, as they could have left the cursor near the distant targets when using direct touch to select nearby targets. However, this can create usability issues if the user is going to be interacting with distant objects in multiple directions, or if they lose their cursor and are having difficulty locating it. A special gesture, such as in Gillot et al. [8], could be implemented to return the cursor to the user's location.

A promising extension of this method that could warrant deeper investigation is a hybrid of absolute and cursor- based input, where users can use the cursor to select a broad group of controls, creating a local copy of those controls at their current position. This target could be made much larger, and would therefore be significantly easier to select, allowing for users to interact with the controls inside the group more comfortably with absolute input. This could conceivably combine the speed and multi-user benefits of the long-distance hybrid method with the accuracy and familiarity of interacting with more complex controls using direct touch input directly in front of them.

While the HybridCursor Touch technique is implemented for multi-touch displays, use of the cursor itself does not necessarily rely on multi-touch; a gesture such as a drawn circle or a double-tap could instead be used to spawn the cursor, and another gesture such as a double-tap could trigger a selection at the location of the cursor. This would allow the technique to be implemented for inexpensive touch displays that only support single touch, or which are less reliable when used with many simultaneous touches, such as the apparatus used in our experiments in a multi-user setting.

The filtering techniques we implemented proved helpful in improving the stability of the infrared optical touchscreen; however, the input was still not optimally stable. A problem that participants occasionally encountered was a touch suddenly jumping somewhere else on the display; these high-speed jumps would be affected by the sigmoid CD gain curve and could move the cursor or the screen a long distance. Future improvements on the filtering transformation could include detecting an instantaneous jump across a long distance, and attempting to cancel that movement in such a way that still creates continuity of input for the user's finger.

Occasionally, the participant's finger would cease to be detected by the optical frame for a period of time that was too long to simulate continuity of input without causing excessive false positives, causing discrete touches to be interpreted as movement. This was especially problematic in the multi-user setting, as the screen's sensors are only at the edges and thus are vulnerable to one participant shadowing the fingers of the other participant. It is not clear how this could be prevented, other than eventually replacing these optical touch frames with a 2-dimensional grid of sensors that isn't vulnerable to shadowing.

Chapter 9

Conclusions

Our distance interaction method is not prohibitively slower than existing input methods for absolute and relative input on large touch screens at close distances, and at long distances well exceeds them in speed, and is comparable in accuracy. Our method provides important ergonomic benefits over existing methods, such as a reduced need for movement to interact with distant targets, as well as the ability for multiple users to perform relative input interactions simultaneously. Multiple users can successfully use the interaction technique, and willingly choose to use it to interact with both near and far targets for multiple types of input.

The use of two-finger drag with a similar cursor acceleration function provides moderate improvements in target selection speed and large improvements in accuracy. While this interaction method is not suitable for multi-user interaction, it remains a feasible choice for single-user interaction on large touchscreens.

References

[1] Azad, A., Ruiz, J., Vogel, D., Hancock, M., and Lank, E. Territoriality and behaviour on and around large vertical publicly-shared displays. In *Proceedings of the Designing Interactive Systems Conference* (New York, NY, USA, 2012), DIS '12, ACM, pp. 468–477.

[2] Ballagas, R., Borchers, J., Rohs, M., and Sheridan, J. G. The smart phone: a ubiquitous input device. *IEEE Pervasive Computing 5*, 1 (2006), 70–77.

[3] Banerjee, A., Burstyn, J., Girouard, A., and Vertegaal, R. Pointable: An in-air pointing technique to manipulate out-of-reach targets on tabletops. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces* (New York, NY, USA, 2011), ITS '11, ACM, pp. 11–20.

[4] Baudisch, P., Cutrell, E., Robbins, D., Czerwinski, M., Tandler, P., Bederson, B., Zierlinger, A., and others. Drag-and-pop and drag-and-pick: Techniques for accessing remote screen content on touch-and pen-operated systems. In *Proceedings of INTERACT* (2003), vol. 3, pp. 57–64.

[5] Casiez, G., Roussel, N., and Vogel, D. 1 € filter: A simple speed-based low-pass filter for noisy input in interactive systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2012), CHI '12, ACM, pp. 2527–2530.

[6] Fitts, P. M. The information capacity of the human motor system in controlling the amplitude of movement. In *Journal of Experimental Psychology, Vol 47(6)* (1954), APA, pp. 381–391.

[7] Forlines, C., Vogel, D., and Balakrishnan, R. HybridPointing: fluid switching between absolute and relative pointing with a direct input device. In *Proceedings of*

the 19th annual ACM symposium on User interface software and technology, ACM, pp. 211–220.

[8] Gilliot, J., Casiez, G., and Roussel, N. Direct and indirect multi-touch interaction on a wall display. In *Proceedings of the 26th Conference on L'Interaction Homme-Machine* (New York, NY, USA, 2014), IHM '14, ACM, pp. 147–152.

[9] Jakobsen, M. R., Jansen, Y., Boring, S., and Hornbæk, K. *Should I Stay or Should I Go? Selecting Between Touch and Mid-Air Gestures for Large-Display Interaction*. Springer International Publishing, Cham, 2015, pp. 455–473.

[10] Khan, A., Fitzmaurice, G., Almeida, D., Burtnyk, N., and Kurtenbach, G. A remote control interface for large displays. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology* (New York, NY, USA, 2004), UIST '04, ACM, pp. 127–136.

[11] Kim, S., Yu, J., and Lee, G. Interaction techniques for unreachable objects on the touchscreen. In *Proceedings of the 24th Australian Computer-Human Interaction Conference* (2012), ACM, pp. 295–298.

[12] Liu, M., Nancel, M., and Vogel, D. Gunslinger: Subtle arms-down mid-air interaction. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology* (2015), ACM, pp. 63–71.

[13] Microsoft. Pointer ballistics for windows xp, 2002.

[14] Nancel, M., Pietriga, E., Chapuis, O., and Beaudouin-Lafon, M. Mid-air pointing on ultra-walls. *ACM Transactions on Computer-Human Interaction (TOCHI) 22*, 5 (2015), 21.

[15] Parker, J. K., Mandryk, R. L., and Inkpen, K. M. TractorBeam: seamless integration of local and remote pointing for tabletop displays. In *Proceedings of Graphics interface 2005* (2005), Canadian Human-Computer Communications Society, pp. 33–40.

[16] Reetz, A., Gutwin, C., Stach, T., Nacenta, M., and Subramanian, S. Superflick: a natural and efficient technique for long-distance object placement on digital tables. In *Proceedings of Graphics interface 2006* (2006), Canadian Information Processing Society, pp. 163–170.

[17] Shneiderman, B. Touch screens now offer compelling uses. *IEEE software 8*, 2 (1991), 93–94.

[18] Su, Q., Au, O. K.-C., Xu, P., Fu, H., and Tai, C.-L. 2d-Dragger: unified touch-based target acquisition with constant effective width. ACM Press, pp. 170–179.

[19] Wallace, J. R., Vogel, D., and Lank, E. The effect of interior bezel presence and width on magnitude judgement. In *Proceedings of Graphics Interface 2014* (2014), Canadian Information Processing Society, pp. 175–182.

[20] Wobbrock, J. O., Findlater, L., Gergle, D., and Higgins, J. J. The Aligned Rank Transform for Nonparametric Factorial Analyses Using Only Anova Procedures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2011), CHI '11, pp. 143–146.

Appendix A: Filtering

As the apparatus uses infrared sensors around the edge of the frame to detect touches, the input can be very noisy, particularly when the user has many fingers near the frame. In cases where the user's fingers create an ambiguous situation, the driver attempts to guess what the correct configuration of touches is. While the driver does some amount of filtering of its own, additional filtering was required to remove flickering that could be detected as finger taps or the number of held fingers changing.

Filtering was performed by introducing an intermediate layer between the driver's reported touch events and the touch events that were presented to the rest of the program. This separate layer presents its own touch data structures, allowing full control over the position and number of active touches.

Filtering consists of several stages:

1. **All touches coming from the driver that are too close to the edge of the screen are ignored.** Targets in the task are intentionally not placed near the edge of the screen, and the sensors occasionally detected spurious touches stuck to the edges. The radius used was 64 pixels, or 3.5 cm.

2. **All new touches coming from the driver are not presented to the program for a short period of time.** If a touch from the driver is lifted within 20 ms, it is considered a spurious touch and is discarded without ever being presented to the rest of the program. If the touch is within 64 pixels, or 3.5cm, of an existing non-spurious touch, the touch is "absorbed" into the existing touch (see below). If the touch is within 64 px, or 3.5cm, of a non-spurious touch which was lifted within the last 40 ms, the new touch is considered to be a continuation of the recently-lifted touch, and no touch-up or touch-down events are sent to the rest of the program.

3. **All lifted touches coming from the driver do not send a touch-up event to the program for a short period of time.** The 40 ms pause mentioned above allows the touch to be continued, ignoring spurious gaps in touch data caused by the user's finger slightly exiting the frame's sensors. If the 40 ms delay passes without the lifted touch being continued, then a touch-up event is sent to the rest of the program as usual.

4. **When a touch from the driver is absorbed into an existing touch, the movement of that finger is ignored for the purposes of moving the touch.** Only the first finger that triggered the touch actually controls how the touch moves. However, if the absorbed finger moves far enough away from the initial finger's current position, the absorbed finger will split off into a new touch, which is treated identically to a touch for which the driver just sent a touch-down event, as in item 2. If the initial finger for the touch is lifted, then the next absorbed finger takes over as the finger that controls the touch. When all absorbed fingers are lifted, we wait to send a touch-up event as in item 3.

5. **A touch tap event is triggered if a non-spurious touch is touched down and touched up within a short period of time, across a small distance.** A "tap" consists of a non-spurious touch that exists for at most 400 ms. Because of the small delays introduced by the filtering process, this means that short, brisk swipes might be seen as "taps", so to help discriminate, any touch that moves more than 64 px, or 3.5 cm, total during its lifetime will not trigger a tap.

6. **The movement of a non-spurious touch is lowpass-filtered.** Having a finger that is controlling cursor motion or screen motion jump around can be very frustrating when trying to select small or distant targets. We apply the 1€ filter [5] to the coordinates of the touch movements that are presented to the rest of the program. We use the parameters 1.0 for *minimum cutoff*, 0.007 for *beta*, and 1.0 for *derivative cutoff*.

This filter was active during every interaction mode in both experiments. The parameters were identical for each interaction mode, except for one: as the *CursorTap* technique is particularly sensitive to spurious touch-taps on the finger that moves the cursor, and there is generally only one finger moving the cursor, we expand the radius in which a touch can be considered a continuation of a recently lifted touch. The radius is set to the maximum of the default (64 px, or 3.5 cm) and 1.1 times the distance moved between the finger's current position and the finger's last position, such that if the finger disappears from the frame then reappears while moving at a constant velocity, it will

continue the previous touch instead of triggering a touch-down and a touch-up. This expansion happens only for the finger that is moving the cursor.