Current Topics in Media Computing and HCI

Experimental Research Part 2 + Reviews in HCI

Prof. Dr. Jan Borchers
Media Computing Group
RWTH Aachen University

Summer Term 2020

https://hci.rwth-aachen.de/cthci
Experimental Variables: Measurement Scales

- **Nominal scale:** discrete, qualitative, categorical differences, ignoring the order
  - E.g., input techniques: mouse vs. touchscreen (IV), whether the user made an error or not (DV)

- **Ordinal scale:** sequentially ranked categories, ignoring magnitude of differences
  - E.g., small/medium/large keyboard buttons (IV), Likert (5-point) scale answers (DV)

- **Interval scale:** sequentially organized categories, all categories have the same size (possible to determine relative distances), but no ratios; e.g., temperature (IV, DV)

- **Ratio scale:** interval scale in which zero represents complete absence (possible to determine absolute distances and thus ratios)
  - E.g., Task completion time in seconds (DV), error rate in percent (DV)
Basic Experimental Designs

Between-groups design

Within-groups design
Basic Experimental Designs

• Between-groups design
  • Each subject only does one variant of the experiment
  • There are at least 2 groups to isolate effect of manipulation:
    • Treatment group and control group
    • Advantage: no practice effects across variants
    • Disadvantage: requires more users
  • Good for tasks that are simple and involve limited cognitive processes, e.g., tapping or visual search

• Within-groups design
  • Each subject does all variants of the experiment
  • Advantage: Fewer users required, individual differences canceled out
  • Disadvantage: practice effects may occur
  • Good for complex tasks, e.g., typing, reading, composition, problem solving
Order Effects

• Problem of within-groups design

• Behavior may be influenced by experience that occurred earlier in the sequence

• **Carryover effects:** changes caused by the lingering aftereffects of an earlier treatment condition
  
  • E.g., testing the first condition causes users’ fingers to hurt, degrading their performance in the second condition

• **Progressive error:** changes that are related to general experience in the study but unrelated to specific treatments
  
  • Practice effects and fatigue
  
  • E.g., the experiment takes too long overall
Counterbalancing

• Tries to negate order effects

• Ideally: Use every possible order of treatments with an equal number of individual participants

• Compromise: Latin Square
  • Each condition appears at each ordinal position
  • Each condition precedes and follows each condition once
  • Example: six treatments A, B, C, D, E, F
Learning Curve

• **Learning curve**: relationship between experience (or time) and performance

• Rapid raise at the beginning, followed by a plateau

• In general, start measuring when the learning effect is gone!
In-Class Exercise #1

• You have designed a new keyboard layout, and you want to know how good it is

• Strategy: compare it to existing techniques

• Describe one reason to choose a
  • Within-groups design
  • Between-groups design
Experimental Design In-class Exercise #1

- Usually preferred: *within-groups design*
  - Minimizes confounding effects from the behavioral differences between participants

- Sometimes, we need a *between-groups* design
  - E.g., when testing whether a keyboard favors users with right-handedness over those with left-handedness
  - When there are interferences between conditions, e.g., different keyboard layouts on the same hardware
In-class Exercise #2: Experimental Study Basics

Confounding variables
- Manipulate
- Measure
- Compare

Validity:
- Internal
- External

Scales:
- Nominal
- Categorical
- Interval
- Ratio

Hold constant
- Matching
- Random assignment

Controls

Learning curve
Counterbalancing
Latin Square
Order effects

Extraneous variables
Confounding variables
In-class Exercise #2: Experimental Study Basics

**Manipulate**

**Measure**

**Compare**

**Controls**

**Scales:**
- Nominal
- Categorical
- Interval
- Ratio

**Order effects**

**Validity:**
- Internal
- External

**Counterbalancing**

**Confounding variables**

**Learning curve**

**Order effects**

**Hold constant**
- Random assignment

**Latin Square**

**Matching**

**Random assignment**

**Extraneous variables**

**Learning curve**

**Validity**

**Hold constant**
- Random assignment
CHAPTER 14

Publishing HCI Research
Criteria for a Good Paper

- **Contribution**: What new insight does it bring to the field?
- **Benefits**: What can one learn from this / do with this?
- **Novelty**: Prior publications?
- **Validity**: Are the claims properly backed up?
- **Applicability**: How good does the paper match the likely audience?
- **Format**: Readability, consistency and clarity
  - Clear presentation of the content, figures, graphs, other visuals etc.
Recap: Validity

- **Internal validity**: a single, unambiguous explanation for the relationship between two variables
  - Threats: e.g., confounding variables, experimenter bias, learning effect, Hawthorne effect (being observed causes the changes)

- **External validity**: extent to which we can generalize the results to people, settings, times, measures, and characteristics other than those used in that study
  - Threats: e.g., generalizing across participants, multiple IVs interference

- Always a trade-off, strike an appropriate balance depending on the goal of your research
Originality

- What new ideas or approaches are introduced?

- **Very important**: an acceptable paper must make a clear contribution to Human–Computer Interaction
Guided Paper Walkthrough
Figure 1: The user wants to tap the button in the top left corner of her smartphone screen to select all images at once. Reaching this button, however, is hardly possible without changing the device grip, which leads to increased device motion and thus is likely to result in a device drop. The ForceRay interaction technique addresses this reachability issue without destabilizing the device grip: (a) When the user applies a force touch on the screen, a ray is displayed that extends the user's thumb reach to the opposing screen edge. (b) The user drags her thumb to reposition the ray until it crosses the desired target. (c) To reach the button, the user increases her force to move the red cursor along the ray. The cursor automatically highlights the first selectable target, an image, on the ray. The more force is applied, the farther the cursor moves along the ray, always highlighting the underlying target. (d) At maximum force, the 'Select All' button is reached. (e) Lifting the thumb triggers the button.
Figure 1: The user wants to tap the button in the top left corner of her smartphone screen to select all images at once. Reaching this button, however, is hardly possible without changing the device grip, which leads to increased device motion and thus is likely to result in a device drop. The ForceRay interaction technique addresses this reachability issue without destabilizing the device grip: (a) When the user applies a force touch on the screen, a ray is displayed that extends the user’s thumb reach to the opposing screen edge. (b) The user drags her thumb to reposition the ray until it crosses the desired target. (c) To reach the button, the user increases her force to move the red cursor along the ray. The cursor automatically highlights the first selectable target, an image, on the ray. The more force is applied, the farther the cursor moves along the ray, always highlighting the underlying target. (d) At maximum force, the ‘Select All’ button is reached. (e) Lifting the thumb triggers the button.
ABSTRACT

Smartphones are used predominantly one-handed, using the thumb for input. Many smartphones, however, have grown beyond 5". Users cannot tap everywhere on these screens without destabilizing their grip. ForceRay (FR) lets users aim at an out-of-reach target by applying a force touch at a comfortable thumb location, casting a virtual ray towards the target. Varying pressure moves a cursor along the ray. When reaching the target, quickly lifting the thumb selects it. In a first study, FR was 195 ms slower and had a 3% higher selection error than the best existing technique, BezelCursor (BC), but FR caused significantly less device movement than all other techniques, letting users maintain a steady grip and removing their concerns about device drops. A second study showed that an hour of training speeds up both BC and FR, and that both are equally fast for targets at the screen border.

CCS CONCEPTS

- Human-centered computing → Gestural input;

KEYWORDS

Touch; force; pressure; mobile; one-handed; reachability

ACM Reference Format:
Abstract

Smartphones are used predominantly one-handed, using the thumb for input. Many smartphones, however, have grown beyond 5". Users cannot tap everywhere on these screens without destabilizing their grip. ForceRay (FR) lets users aim at an out-of-reach target by applying a force touch at a comfortable thumb location, casting a virtual ray towards the target. Varying pressure moves a cursor along the ray. When reaching the target, quickly lifting the thumb selects it. In a first study, FR was 195 ms slower and had a 3% higher selection error than the best existing technique, BezelCursor (BC), but FR caused significantly less device movement than all other techniques, letting users maintain a steady grip and removing their concerns about device drops. A second study showed that an hour of training speeds up both BC and FR, and that both are equally fast for targets at the screen border.

CCS CONCEPTS
- Human-centered computing → Gestural input;

KEYWORDS
Touch; force; pressure; mobile; one-handed; reachability

ACM Reference Format:
1 INTRODUCTION

Smartphone touchscreens have been growing in size [16], from 3.5” in Apple’s original iPhone from 2007 to 6.5” in their current iPhone Xs Max, for example. While larger screens can show more content at a time, they are a mixed blessing for touch input. Users tend to interact with their smartphones using a single hand, whether due to user preference [3, 20] or because the other hand is holding a coffee cup, carrying a bag, or holding on during a train ride. This leaves the thumb as the only finger to interact with the touchscreen [25]. This way, however, the user cannot comfortably reach all parts of the screen unless she re-grasps the device, which is inconvenient and takes time. Most critically, re-grasping destabilizes the device grip and causes increased device motion. This makes users feel insecure in holding their device [13, 15] and can lead to accidental drops, breaking the screen or other components. This out-of-reach area grows with screen size.

Industry and HCI research have proposed several ways to mitigate this reachability issue (see Related Work), but these approaches require explicit mode switching, allow using only a small part of the screen for interaction, or still cause significant device motion while selecting targets.

We present ForceRay (FR), a reachability technique that addresses these issues. It uses the force sensing touchscreen found in recent smartphones: When the user applies a force touch with her thumb on the touchscreen, a ray appears that points from the lower screen corner under her palm through her thumb’s touch position to the opposite edge of the screen (Fig. 1). If necessary, she can roll her thumb left or right to fine-tune the direction of the ray so that it crosses the intended target. Along this virtual thumb extension, she controls a cursor: The more force she applies, the further the cursor moves away from the thumb. If the cursor exits a target, the next target along the ray is highlighted automatically. To select it, the user quickly lifts her thumb off the screen. To cancel instead, she reduces her force below the force touch activation threshold before lifting the finger.

FR is a “quasi-mode” [37]. It is active only while the user is consciously maintaining her force touch, thus avoids confusing and time-consuming mode switching through other explicit input gestures. Furthermore, it benefits natural, ergonomic thumb movement that enables the user to maintain a stable device grip. FR’s design scales to different screen sizes and form factors, and makes it especially fast to select targets at screen borders by simply applying maximum force.

Thus, the key contribution of this paper is the FR interaction technique that extends thumb reach via force input to enable selection of out-of-reach targets with a steady device grip. In the remainder of this paper, we first review related work before describing the design and implementation of FR. To validate FR, we present two user studies: Study 1 compared FR to standard direct touch input and three existing reachability techniques: the One-Handed Mode, found in Android devices, that downscales and moves the screen towards the user’s thumb. MagStick [39], and BeezICursor (BC) [29]. Among all, FR significantly caused the least device motion. Study 2 showed that an hour of training sped up FR selection time and that users selected far targets at the screen border as fast as the fastest candidate from Study 1, BC, with 96% selection accuracy. We close with recommendations to address reachability issues for one-handed touchscreen use.
1 INTRODUCTION

Smartphone touchscreen have been growing in size [16], from 3.5" in Apple’s original iPhone from 2007 to 6.5" in their current iPhone Xs Max, for example. While larger screens can show more content at a time, they are a mixed blessing for touch input. Users tend to interact with their smartphones using a single hand, whether due to user preference [3, 20] or because the other hand is holding a coffee cup, carrying a bag, or holding on during a train ride. This leaves the thumb as the only finger to interact with the touchscreen [25]. This way, however, the user cannot comfortably reach all parts of the screen unless she re-grasps the device, which is inconvenient and takes time. Most critically, re-grasping destabilizes the device grip and causes increased device motion. This makes users feel insecure in holding their device [13, 15] and can lead to accidental drops, breaking the screen or other components. This out-of-reach area grows with screen size.

Industry and HCI research have proposed several ways to mitigate this reachability issue (see Related Work), but these approaches require explicit mode switching, allow using only a small part of the screen for interaction, or still cause significant device motion while selecting targets.

We present ForceRay (FR), a reachability technique that addresses these issues. It uses the force sensing touchscreen found in recent smartphones. When the user applies a force touch with her thumb on the touchscreen, a ray appears that points from the lower screen corner under her palm through her thumb’s touch position to the opposite edge of the screen (Fig. 1). If necessary, she can roll her thumb left or right to fine-tune the direction of the ray so that it crosses the intended target. Along this virtual thumb extension, she controls a cursor: The more force she applies, the further the cursor moves away from the thumb. If the cursor exits a target, the next target along the ray is highlighted automatically. To select it, the user quickly lifts her thumb off the screen. To cancel instead, she reduces her force below the force touch activation threshold before lifting the finger.

FR is a "quasi-mode" [37]. It is active only while the user is consciously maintaining her force touch, thus avoids confusing and time-consuming mode switching through other explicit input gestures. Furthermore, it benefits natural, ergonomic thumb movement that enables the user to maintain a stable device grip. FR’s design scales to different screen sizes and form factors, and makes it especially fast to select targets at screen borders by simply applying maximum force.

Thus, the key contribution of this paper is the FR interaction technique that extends thumb reach via force input to enable selection of out-of-reach targets with a steady device grip. In the remainder of this paper, we first review related work before describing the design and implementation of FR. To validate FR, we present two user studies: Study 1 compared FR to standard direct touch input and three existing reachability techniques: the One-Handed Mode, found in Android devices, that downscales and moves the screen towards the user’s thumb. MagStick [39], and BuzzCursor (BC) [29]. Among all, FR significantly caused the least device motion. Study 2 showed that an hour of training sped up FR selection time and that users selected far targets at the screen border as fast as the fastest candidate from Study 1, BC, with 95% selection accuracy. We close with recommendations to address reachability issues for one-handed touchscreen use.
2 RELATED WORK

Professor Jan Borchers: Current Topics in Media Computing and HCI

Related Work

Contribution?

Benefit?

Originality?

Validity?

Novelty?

Format?

To make target highlighting more efficient, FR does not require the user to move the cursor precisely onto the target. Since the targets the ray crosses follow a defined sequence, which the cursor will reach the next target, the next target can already be highlighted before actually entering it (Fig. 1 c–d). In addition, the segment to the first target on the ray is associated with the first target, and the ray segment beyond the last target is associated with the last target. Such virtual cursor enlargement is similar to the idea behind techniques like Bubble Cursor [18] or Dynaplot [8]. Cursor enlargement not only speeds up the highlighting process, but also avoids invalid selections. To finally select the highlighted target, the user quickly lifts her thumb (Fig. 1 e). This selection mechanism, also known as Quick Release, is a common technique to confirm a selection with a single mouse click and is used in various applications.
2 RELATED WORK

PR addresses reachability and force input techniques, both on mobile devices. We discuss the related work successively.

Reachability Techniques

Probably the most straightforward approach is to constrain the UI layout to just the comfort region of the thumb. With Droid almost in mind, Borchers et al. [2] built a model that predicts where the user’s thumb can reach. However, this limits the space for interactive elements to a constant area in one screen corner, ignoring the extra available space up beyond.

To survey solutions that address reachability for UIs laid out on the entire screen, Chang et al. [7] constructed a design space that classifies interactions by their trigger and targeting mechanisms. Trigger mechanisms look at how the technique is activated, and targeting mechanisms address how a target is selected. The latter distinguish between techniques that apply a screen transform, provide a proxy region, and use a cursor to select a target. We follow this useful taxonomy.

Screen Transform Techniques. Smartphone manufacturers embed such techniques in the mobile OS. In iOS, double-tapping the Home button or swiping down across the bottom screen edge slides the screen half down, but this leaves targets on the far side opposite of the thumb unreachable. Samsung’s One-Handed Mode shrinks the entire screen to be close to the thumb when triple-tapping the Home button or sliding from the corner. TiltReduction [7] shrinks the screens likewise when tilting the device. SlidingScreen [26] moves the screen diagonally towards the thumb, and is either triggered by swiping or by generating a large touch footprint. TiltSlide [7] works similarly, but is activated by tilting the device. MovingScreen [45] also moves the screen to the thumb, depending on how far the user swipes from the screen edge. Le et al. [28] trigger the same effect by sliding the index finger across a touchscreen at the back of the device (Bionic). Lochtiedel et al. [31] detect which hand activated the device to shift the UI towards that hand. Eardley et al. [13, 14] present several adaptive UI concepts, e.g., a keyboard that slides along with the thumb.

All these techniques, however, either omit parts of the UI and thus hide context information, or shrink targets, making targets more difficult to hit, or need hardware modifications, or they use tilt, which is prone to overshooting and making reading the screen difficult at certain angles [42].

Proxy Region Techniques. TapJar [39] magnifies a part of the screen around the thumb’s touch location in a pop-up view. ThumbSpace [28, 26] is a similar concept but here, the view represents the entire screen, making targets very small and difficult to hit. Both techniques do not scale well to large form factors and their proxy views obscure a part of the screen. Hasan et al. [19] proposed the mid-air space between thumb and touchscreen as proxy region. Lochtiedel et al. [30] used a B2D touchpad to reach upper targets with the index finger from behind. Such proxy region extends the thumb’s reach by 15% [36]. However, these techniques require hardware modifications.

Cursor Techniques. ThumbCursor [7] lets users drag an accelerated cursor with the thumb and is triggered by tilting. ThumbCursor [29] is similar but activated by swiping from the bezel, which overwrites system-wide triggers, e.g., for opening the phone’s control panel. ExtendedThumb [27] is a faster cursor triggered by a double tap and—depending on where the cursor is selected—extends the thumb with a cursor whose offset increases with drag speed. ExtendedCursor [26] is similar but slides the cursor opposite to where the user drags her thumb. MagicStick [39] also uses this opposite dragging mechanism to avoid occluding the target with the thumb and is triggered when pressing on the screen. However, the swiping that these techniques require, can lead to grip instability when the thumb is moved beyond its comfort region. 2D-Dragger [44] solves this problem by stepwise pushing the UI elements located in the lower right (left) screen corner for right (left) handed users. Usually, the user will initiate the drag at a point that makes the thumb already go through the intended target. If she misses, she can simply move her thumb to the left or right to reposition the ray until it intersects with the target. 3D-Dragger [44] problem, the further the user moves upwards on the ray, the closer the user controls via force input to highlight any target that intersects with the ray. The more force is applied, the further the cursor moves upwards on the ray towards the opposite screen edge (Fig. 1 c–d). To make this mapping consistent, i.e., have the same stroke length in force always result in the same cursor travel distance, PR maps the available force range between force threshold and maximum possible force to the length of the touchscreen diagonal, which is the longest possible ray length. A force mapping that always maps the same force to the same value, known as position linearity, would be [40] and allows for instant corrections to over- and undershoots by simply decreasing or increasing the force.

To make this mapping consistent, i.e., have the same stroke length in force always result in the same cursor travel distance, PR maps the available force range between force threshold and maximum possible force to the length of the touchscreen diagonal, which is the longest possible ray length. A force mapping that always maps the same force to the same value, known as position linearity, would be [40] and allows for instant corrections to over- and undershoots by simply decreasing or increasing the force.

To make this mapping consistent, i.e., have the same stroke length in force always result in the same cursor travel distance, PR maps the available force range between force threshold and maximum possible force to the length of the touchscreen diagonal, which is the longest possible ray length. A force mapping that always maps the same force to the same value, known as position linearity, would be [40] and allows for instant corrections to over- and undershoots by simply decreasing or increasing the force.

To make this mapping consistent, i.e., have the same stroke length in force always result in the same cursor travel distance, PR maps the available force range between force threshold and maximum possible force to the length of the touchscreen diagonal, which is the longest possible ray length. A force mapping that always maps the same force to the same value, known as position linearity, would be [40] and allows for instant corrections to over- and undershoots by simply decreasing or increasing the force.

To make this mapping consistent, i.e., have the same stroke length in force always result in the same cursor travel distance, PR maps the available force range between force threshold and maximum possible force to the length of the touchscreen diagonal, which is the longest possible ray length. A force mapping that always maps the same force to the same value, known as position linearity, would be [40] and allows for instant corrections to over- and undershoots by simply decreasing or increasing the force.

To make this mapping consistent, i.e., have the same stroke length in force always result in the same cursor travel distance, PR maps the available force range between force threshold and maximum possible force to the length of the touchscreen diagonal, which is the longest possible ray length. A force mapping that always maps the same force to the same value, known as position linearity, would be [40] and allows for instant corrections to over- and undershoots by simply decreasing or increasing the force.

To make this mapping consistent, i.e., have the same stroke length in force always result in the same cursor travel distance, PR maps the available force range between force threshold and maximum possible force to the length of the touchscreen diagonal, which is the longest possible ray length. A force mapping that always maps the same force to the same value, known as position linearity, would be [40] and allows for instant corrections to over- and undershoots by simply decreasing or increasing the force.

To make this mapping consistent, i.e., have the same stroke length in force always result in the same cursor travel distance, PR maps the available force range between force threshold and maximum possible force to the length of the touchscreen diagonal, which is the longest possible ray length. A force mapping that always maps the same force to the same value, known as position linearity, would be [40] and allows for instant corrections to over- and undershoots by simply decreasing or increasing the force.

To make this mapping consistent, i.e., have the same stroke length in force always result in the same cursor travel distance, PR maps the available force range between force threshold and maximum possible force to the length of the touchscreen diagonal, which is the longest possible ray length. A force mapping that always maps the same force to the same value, known as position linearity, would be [40] and allows for instant corrections to over- and undershoots by simply decreasing or increasing the force.
STUDY: REACHABILITY TECHNIQUES

To understand how FR compares to the state of the art, we conducted a user study with 15 participants (21–33 years, \( M = 25.75 \), SD = 3.1; two female, all right-handed; thumb length: \( M = 74.87 \) mm, SD = 7.12 mm). They were all smartphone users (screen size: \( M = 5.86 \), SD = 0.46). We compared FR to One-Handed Mode (OM, similar to Samsung’s) and BezelCursor [39], and BezelCursor (BC) [29] to cover a good variety of techniques. OM is a common commercial solution, MS is one of the first mobile reachability techniques and is often compared to by other papers, and BC is well scalable. We added Direct Touch (DT) input as baseline. We asked users to select targets within the screen without moving their thumb when using each of these techniques while holding the device in their right hand in portrait orientation.

Apparatus, Techniques, and Task

Players were asked to answer the task to our users and to capture data. For our implementation of FR, we used the force readings provided by the force-sensitive touchscreen. According to Apple’s documentation [21], the force sensor API delivers unlimited force values between \( 0 \) and \( 65535 \) steps of 2, with force sensitivity set to “Fine”. Values around 1.0 should be interpreted as an ordinary touch; higher values as intentional input force. Although Apple does not state how these values translate to Newtons, experiments [34] suggest a 4 N range and a linear transfer function. Although FR combines dragging while exerting force, friction is low due to the small 4 N force range and the smoothness of the touchscreen glass surface. The iPhone screen measured 736/1441 pt. For iPhone 6/7/8/Plu, 1 pt = 1 mm.

Techniques: One-Handed Mode (OM) mimics Samsung’s reachability technique: by default, it deactivates the UI to 2/3 of its original size and moves it to the lower right corner (Fig. 2, center). Unlike Samsung’s trigger, that is either a swipe from the corner or a tap on the Home button, we show OM, double tap gesture for activation. Swiping could have been confusing with BC trigger, and pilot tests revealed that users found triple taps confusing to perform. Since the iOS SDK does not notify a target or touch events on the Home button, users instead double-tapped directly above the Home button on a pink button on the touchscreen. The UX shrinking and movement animation duration was 263 ms. MagStick (MS) [39] is triggered by force-touching on the screen using the same threshold as used for FR. To highlight a target, the user drags her thumb, which displays a line that grows in the opposite direction of where the thumb is moved. The line has two components of the same length, with the center located at where the thumb was initially placed. When the upper component gets within 5 mm of a target, it is highlighted, and the line snaps to its center as if magnetized. Lifting off the thumb selects the target.

BezelCursor (BC) [29] is triggered by swiping from the bezel towards the touchscreen. This displays a line that grows linearly by a factor of three in the direction of the thumb. The line ends at a circular cursor that expands exponentially to 7.5 mm depending on how fast the user swipes her thumb. This area cursor is equivalent to DynamicSpot [8], and shrinks co-exponentially when swiping acceleration falls below 2 m/s². If a target is intersected by the cursor, it is highlighted. When multiple targets are displayed, the user needs to reach the smallest distance from the center to the cursor location is chosen. Lifting off the thumb selects the target.

ForceRay (FR) implements the interaction design described earlier. We set the force threshold to 1.15 units, which is an ordinary touch. We implemented the Quick Release mechanism for confirming the selection of a highlighted target as described in [35] since this was the default method. Pilot tests revealed that these did not fit all users, leading to increased selection errors. We followed [16] to determine individual timing parameters by calibration trials upfront (\( N = 66–224 \) ms before complete thumb lift-off, \( SD = 52–48 \) ms). Pilot testing also optimized the CMS reference point placement to 3.2 mm away from the right and bottom of the screen, setting it exactly as the lower right corner would not allow users to comfortably move the ray to the right edge. For mapping the force input to the cursor position \( \left( x_{\text{CMS}}, y_{\text{CMS}} \right) \) on the way, we first used a linear transfer function as suggested by [45], but users tended to overshoot targets. Therefore, we designed a logarithmic transfer function whose slope decreases from 95% to 25% from touch force threshold to maximum sensible force \( F_{\text{CMS}} = \left( 4 \ t \left( x_{\text{CMS}} + 5 \right) / 895 \right) + s \), where \( F_{\text{CMS}} \) is the relative force ranging from force touch threshold to maximum sensible force, \( s \) is a scaling factor for the screen size, and \( s \) is in a distance from the ray start to where the cursor should start on the ray. For our setup, we set \( s = 1452 \) pt and \( s = 248 \) pt minus the dynamic distance from the thumb’s touch position to the CMS reference point. Furthermore, to reduce ray and cursor jitter, we filtered touch location and force using the 1N Filter [5] (\( s = 1 \)).

Targets: Targets were arranged on an invisible 6×3 grid. FR is aimed for the large target set. This is the user for the blue target that is crossed by the force Ray. Currently, the target with the green border is selected since it is the next target on the ray beyond the cursor. OM (minimized): Target layout for the small target set. Top: Full layout before double-tapping the virtual home button (pink). Bottom: OM downscaled by 2/3 after double-tapping the pink button. BC: The user is selecting the blue target with BezelCursor. The red dot visualizes the cursor position that is enlarged by the concentric white circle (DynamicSpot area cursor). Targets in the lower right area (thumb background) were not selectable, since here, targets are directly accessible by the thumb.

Methods

The next Technique about 50% of the time with small vs. large targets first. When users were presented a new Technique, they familiarized themselves with the Technique before performing four test trials for the given target size. They then selected the twelve targets, repeated two times, followed by the remaining size for the current Technique, again starting with four test trials. After all 12 × 2 trials were performed, each user did 280 trials in approximately 45 minutes. Dependent Variables were trial completion Time [ms], users’ Success [1/0], i.e., whether they selected the correct target set or not, and the Gesture Footprint caused by the touches on the screen to capture up to where users had to move their thumb. To quantify device motion and grip stability as in [13, 15], Rotation captured device rotation [°] around x, y, and z axes at 60 Hz. After each Technique, users were asked how much they agreed to (i) that they had to reposition their device grip before acquiring a target, (ii) that they maintained a stable grip while selecting a target, and (iii) that the Technique was easy to apply on a 7-point Likert

Figure 2: Targets (numbered) and distractors arranged on an invisible 6×3 grid. FR is aimed for the large target set. The user is for the blue target that is crossed by the force Ray. Currently, the target with the green border is selected since it is the next target on the ray beyond the cursor. OM (minimized): Target layout for the small target set. Top: Full layout before double-tapping the virtual home button (pink). Bottom: OM downscaled by 2/3 after double-tapping the pink button. BC: The user is selecting the blue target with BezelCursor. The red dot visualizes the cursor position that is enlarged by the concentric white circle (DynamicSpot area cursor). Targets in the lower right area (thumb background) were not selectable, since here, targets are directly accessible by the thumb.
force readings provided by the force-sensitive touchsensor. According to Apple’s documentation [21], the force sensor APT delivers unfiltered force values between 0 and \( \sqrt{2} \) kg steps, with force sensitivity set to “firm.” Values around 1.0 should be interpreted as an ordinary, higher value as intended input force. Although Apple does not state how these values translate to Newtons, experiments [34] suggest a 4 kg range and a linear transfer function. Although FR combines dragging while exerting force, friction is low due to the small 4 N force range and the smoothness of the touchscreen glass surface. The iPhone screen measured 736x414 pt. For iPhone 6/5/7/8 Plus, 1 pt = 16 mm.

Techniques. One-Handed Mode (OM) mimics Samsung’s reachability technique: by default, it_deactivates the UI to 2/3 of its original size and moves it to a lower right corner (Fig. 2, center). Unlike Samsung’s trigger that is either a swipe from the corner or a triple tap on the Home button, we show OM: double tap gesture for activation. Swiping could have been confusing with BC’s trigger, and pilot tests revealed that users found triple taps confusing to perform. Since the iOS SDK does not notify the user about the trigger events on the Home button, users instead double-tapped directly above the Home button on a pink button on the touchscreen. The UI shrinking and movement animation duration was 263 ms.

MagiSTick (MS) [39] is triggered by force-touching on the screen using the same threshold as used for FR. To highlight a target, the user drags her thumb, which displays a line that grows into the opposite direction of where the thumb is moved. The line has two components of the same length, with the center located at where the thumb was initially placed. When the upper component gets within 3 arcmin of a target, it is highlighted, and the line snaps to its center as if magnetized. Lifting off the thumb selects the target.

BezelCursor (BC) [29] is triggered by swiping from the bezel towards the touchscreen. This displays a line that grows linearly by a factor of three in the direction of the thumb. The end of the line has a circular cursor that expands exponentially up to 7.5 mm depending on how fast the user swipes her thumb. This area cursor is equivalent to DynaSpot’s OM [8], and shrinks co-exponentially when swiping acceleration falls below 2 \( \text{m/s}^2 \). When a target is intersected by the cursor, it is highlighted. When multiple targets are close, the cursor selects the target with the smallest distance from its center to the cursor location in chosen. Lifting off the thumb selects the target.

ForceRay (FR) implements the interaction design described earlier. We set the force-touch threshold to 1.35 units, which is significantly higher than an ordinary touch. We implemented the Quick Release mechanism for confirming the selection of a highlighted target as described in [35]. We implemented a Sunlight-Touch interaction, where the mouse was randomized. This did not fit all users, leading to increased selection errors. We followed [10] to determine individual timing parameters by calibration trials upfront (\( \text{N} = 48 -224 \text{ ms} \)) before complete thumb lift-off, SD = 32 -44 ms. Pilot testing also optimized the CMC reference point placement to 3.2 mm away from the right and bottom of the screen, putting it exactly where the right lower corner would not allow users to comfortably move the ray to the right edge. ForceRay is a one-finger touch, but we first used a linear transfer function as suggested by [43], but users tended to overshoot targets. Therefore, we designed a logarithmic transfer function whose slope decreases from 95% to 25% from touch force threshold to maximum sensible force \( \text{F}_\text{CMC} = \left( 1 - \text{F} / \text{F}_\text{CMC} \right) + 0.85% \times \text{F} \), where \( \text{F}_\text{CMC} \) is the relative force ranging from force touch threshold to maximum sensible force, \( \text{s} \) denotes a scaling factor for the screen in pixels. \( 0.3 \times \text{F} \) represents the distance from the ray’s start to where the cursor should start on the ray. For our setup, we set \( \text{s} = 1.852 \text{ pt} \) and \( \text{F} = 240 \text{ pt} \) minus the dynamic distance from the thumb’s touch area to the CMC reference point. Furthermore, to reduce ray and cursor jitter, we filtered touch location and force using the 1/4 Filter \( \text{s} = 1 \).

Targets. Targets were arranged on an invisible 6x10 grid (Fig. 2) across an area of 461x870 pt, each cell measured 69.73 pt. The bottom row 4x10 pt of the screen was excluded to obtain cells with whole numbers. We centered targets at the cells and added distractors to other cells. As in [23], distractors were not centered to avoid a regular-looking arrangement. Since reachability techniques are only meant to replace direct input for targets that are out of reach, distractions close to the thumb (bottom right 4x5 cells) were not selectable. Participants were informed about this and a slight background brightness subtly visualized the exact area. For all techniques, lines were drawn in white, the target to select in blue, the cursor in red, and the border of a highlighted target in green. Hence, when the blue target was highlighted by the green border, the user should lift her thumb. After a target was selected, the next trial was shown automatically after 504 ms. For OM, the UI was automatically shifted back right before showing the next trial.

Innovative Independent Variables were TouchLine (TL), OM, MS, BC, and FR. Distractors, that split our twelve targets (Fig. 2) into two rows of 1x3 and 2x3, were placed at the border of the screen vs. the remaining six targets rather located at the Center, and their corresponding Size (small: 30x30 pt (4.8x4.8 mm) and large: 60x60 pt (9.6x9.6 mm)). Size represented typical UI widget sizes, like the height of a battery level stacked graph, arranged vertically on the iOS Home Screen.

We recorded 5 TouchLine x 12 targets x 2 Size x 2 repetition = 240 trials per user. TouchLine was counter-balanced using a Latin Square design, where Size was randomized. Size was also randomized, but we ensured that each user started testing a new TouchLine about 50% of the time with small vs. large targets first. When users were presented a new TouchLine, they familiarized themselves with the TouchLine before performing four test trials for the given target Size. They then selected the twelve targets, repeated two times, followed by the remaining Size for the current TouchLine, again starting with four test trials. After all 12 x 2 trials were performed, the next TouchLine was presented. Including test trials, each user did 280 trials in approximately 45 minutes.

Innovative Dependent Variables were trial completion Time [ms], number of TouchLine of Size [1], whether they selected the correct target or not, and the Gesture Footprint caused by the touches on the screen to capture up to where users had to move their thumb. To quantify device motion and grip stability as in [13, 15], Rotation captured device rotation [\( \text{\circ} \)] around x, y, and z axis. After each TouchLine, users were asked how much they agreed to (i) that they had regularly change their device grip before acquiring a target, (ii) that they maintained a stable grip while selecting a target, and (iii) that the TouchLine was easy to apply on a 7-point Likert Scale.

Figure 2. Targets (numbered) and distractors arranged on an invisible 6x10 grid. FOV: layout for the large target Size condition. The user is aiming for the blue target that is covered by the ForceRay cursor. Currently, the user taps the green border is selected since it is the next target on the ray beyond the cursor. OM (unmarked): Target layout for the small target Size condition. Top: Full layout before double-tapping the virtual home button (pink). Bottom: UI downsized to 2/3 after double-tapping the pink button. BC: The user is selecting the blue target with BezelCursor. The red dot visualizes the cursor position that is enlarged by the concentric white circle (DynaSpot area cursor). Targets in the lower right area (brighter background) were not selectable since these targets are directly accessible by the thumb.

STUDY 1: REACHABILITY TECHNIQUES

To understand how FR compares to the set of user, we conducted a user study with 15 participants (21–33 years, \( \Delta = 25.78, SD = 3.11 \), two female, all right-handed; thumb length: \( M = 73.67 \text{ mm}, SD = 7.12 \text{ mm} \)). They were all smartphone users (screen size: \( M = 5.86, SD = 0.46 \)). We compared FR to One-Handed Mode (OM, similar to Samsung’s), MagiSTick (MS) [39], and BezelCursor (BC) [29] to cover a good variety of techniques: OM is a common commercial solution, MS is one of the first mobile reachability techniques and often compared to other papers, and BC is well scalable. We added Direct Touch (DT) input as baseline. We asked users to select targets with their thumb on a mobile screen using each of these techniques while holding the device in their right hand in portrait orientation.

Apparatus, Techniques, and Task

9 METHODS

Validity

Novelty

Format

Originality

26 Prof. Jan Borchers: Current Topics in Media Computing and HCI
Results

Since we were interested in how users’ performance is different depending on the TouchInQ used, we focus on analyzing this main effect and related interaction effects. We conducted a repeated-measures ANOVA on the log-transformed Time data. For the dichotomous Success data, we ran McNemar and Cochran’s Q tests. For the analysis of Rotation data we followed [3, 35] by summing up the absolute degrees of device motion change around each axis and ran repeated-measures ANOVAs on the log-transformed data. Likert scale data was compared using Friedman tests.

**Fig. 3**: Study 1: Time [ms] (left) and Success [%] (right) by TouchInQ. For each variable, pairs of levels that do not share a letter are significantly different (Time: all p < .001, Success: all p < .05). Whiskers denote 95% CI.

**Fig. 4**: Study 1: Rotation [°] for the x (left), y (middle), and z-axis (right) by TouchInQ. For each variable, pairs of levels that do not share a letter are significantly different (all p < .001). Whiskers denote 95% CI.

**Fig. 5**: Study 1: Rotation [°] for the x (left), y (middle), and z-axis (right) by TouchInQ. For each variable, pairs of levels that do not share a letter are significantly different (all p < .001). Whiskers denote 95% CI. FR caused almost no device movement.

**Fig. 6**: Study 1: Rotation [°] for the x (left), y (middle), and z-axis (right) by TouchInQ x X (*top*) and TouchInQ x Target (bottom). Yellow cells denote significant differences within TouchInQ (all p < .001). Pairs of levels that do not share a letter are significantly different across TouchInQ (all p < .01). No significant differences were found for the z-axis. CI denotes 95% CI. Star had no effect, but Border targets caused more Rotation than Center targets.

**Fig. 7**: Study 1: Gesture footprint by TouchInQ. Blue dots mark where users started placing their thumb, red dots represent dragging, and green dots indicate where users released their thumb. FR had the smallest coherent footprint and followed ergonomic thumb movement.

**Fig. 8**: Study 1: Means and 95% CI for Likert scale responses (1: totally disagree, 7: totally agree) on and ranking data (right) from the questionnaire. For each statement, pairs of levels that do not share a letter are significantly different (all p < .001). BC was preferred over FR, but not significantly more.

**Fig. 9**: Study 1: Means and 95% CI for Likert scale responses (1: totally disagree, 7: totally agree) on and ranking data (right) from the questionnaire. For each statement, pairs of levels that do not share a letter are significantly different (all p < .001). BC was preferred over FR, but not significantly more.

**Fig. 10**: Study 1: Means and 95% CI for Likert scale responses (1: totally disagree, 7: totally agree) on and ranking data (right) from the questionnaire. For each statement, pairs of levels that do not share a letter are significantly different (all p < .001). BC was preferred over FR, but not significantly more.

**Fig. 11**: Study 1: Means and 95% CI for Likert scale responses (1: totally disagree, 7: totally agree) on and ranking data (right) from the questionnaire. For each statement, pairs of levels that do not share a letter are significantly different (all p < .001). BC was preferred over FR, but not significantly more.
Results

Since we were interested in how users’ performance is different depending on the Touchcues used, we will focus our analysis on this main effect and related interaction effects. We conducted a repeated-measures ANOVA on the log-transformed Time data. For the dichotomous Success data, we ran McNemar and Cochran’s Q tests. For the analysis of Rotation data, we followed [15, 16], by summing up the absolute angles of device motion change around each axis and ran repeated-measures ANOVA on the log-transformed data. Likert scale data was compared using Friedman tests.

Touchcues had a significant main effect on Time (F(4,200) = 548.95, p < .001). Tukey HSD post hoc pairwise comparisons were all significant (p < .001) except between OM and MS (Fig. 3, left). As expected, users were fastest with DT (1.153 ms), followed by BC, for which they needed ≈ 3.24 ms longer. FR was the third fastest Touchcues, yet less than 200 ms slower than BC, OM, and MS were close to 2,000 ms. There was also a Touchcues X Target interaction effect (F(12, 600) = 39.02, p < .001). Fig. 4 (top) lists the Tukey HSD post hoc test results. For each Touchcues except FR, users needed significantly more time to select Border targets compared to Center targets. For FR, on the contrary, this was reversed. Touchcues had a significant main effect on Success (F(4) = 81.62, p < .001, Fig. 3, right). Post hoc tests revealed that Success for BC was significantly higher compared to all other Touchcues except FR. Furthermore, FR and DT yielded significantly higher Success than OM and MS. There was also a Touchcues X Site interaction effect (F(4) = 2.03, p = .041). Fig. 4 (bottom) shows the results from the post hoc tests. For small targets, BC yielded significantly higher Success than DT, OM, MS, and FR had significantly higher Success than OM and DT. For large targets, MS had significantly lower Success than all other Touchcues. Only for DT, there was a significant difference for Success comparing both Site sizes. Small targets had 11% lower Success than large targets.

Touchcues had a significant main effect on Rotation around the x-axis (F(4, 200) = 995.81, p < .001), y-axis (F(4, 200) = 778.61, p < .001), and z-axis (F(4, 200) = 563.31, p < .001). For the x- and y-axes, Tukey HSD post hoc tests revealed that all Touchcues were significantly different from each other (Fig. 5). For the z-axis, post hoc tests revealed that differences between DT and OM and between BC and MS were non-significant. All other pairwise comparisons were significantly different (p < .001). In summary, for each angle, FR always caused the lowest device movement. There were also Touchcues X Target interaction effects for the x-axis (F(4, 200) = 4.30, p = .002) and for the y-axis (F(4, 200) = 6.52, p < .001). Fig. 6 lists the results from the Tukey HSD post hoc tests. In general, for each Touchcues, acquiring targets at the Border resulted in more device movement around both, the x- and y-axes, compared to Center targets.

Fig. 7 shows the Gesture Footprint generated by each Touchcues. FR caused the smallest and most coherent footprint, and touches stayed within the thumb’s comfortable reach, forming natural rotation around the CMC joint.
Discussion

Overall, DT was fastest but achieved low Success for small targets, matching previous findings (e.g., from [20, 24]). DT also caused the strongest device motion, since especially at extremely far positions, like the upper and lower left corner, users had to change their grip, for which some participants almost accidentally dropped the phone, matching findings from [13]. This was why users preferred DT the least. Surprisingly, OM caused the second highest device motion.

Time for OM was highest, reaching almost 2,000 ms. This could be due to the double tap trigger, which, unlike BC and FR, does not contribute to the target selection process, and due to the 200 ms animation time that helps the user understand how the UI is transformed. Both take additional time. Success for OM was also low, especially for small targets (Fig. 4; bottom: A.11:15). Shaking them makes them too hard to hit precisely due to the thumb’s fat finger problem [40]. A solution like AngleFove [38] that captures the finger’s orientation, could help gaining touch precision.

MS had the highest task completion time and lowest Success. Participants found MS more demanding than other techniques, because it required good planning upfront. To reach the top left corner, e.g., the user must place her thumb at least above the center of the screen, so that dragging it to the lower right corner will move the cursor far enough in the opposite direction. When the thumb is placed improperly, it is not possible to correct this within a trial, which explains MS’s low Success, for both, large and small targets. Using the model from [2], we found that for most of our users, the center of the 5.5” screen is not reachable without uncomfortably stretching the thumb or changing the device grip.

Among the reachability techniques, BC was fastest, which was also independent from target size. While users ranked BC best, they remarked that targets at the top were more difficult to reach due to swiping long distances upwards. To ease this, users tilted the device (Fig. 4) towards the thumb, for which they sometimes even had to reposition their grip. Although non-significant, Success for FR was 4% lower than for BC. The Quick Release selection mechanism used in FR could be one explanation, since a lift-off based on force is a less definite event than a lift-off only considering touch information, as used for BC. Unfamiliarity with force control could also explain the lower Success: Cortzen et al. [11] showed that users’ performance for force control significantly increased with training. Our participants reported that selecting close and far targets with FR was fundamentally easier compared to other targets because then they only had to apply minimum or maximum force. Some developed a strategy: Independent of the target location at the border, they first applied maximum force to create a ray with the cursor positioned at its end and then, while maintaining this force, moved the ray onto the target.

Interestingly, some participants remarked that they found FR’s continuous cursor rather a disadvantage and that the discrete green indicator highlighting the preselected target was sufficient. Despite the 1C Filter [5], these users were worried when the cursor would move along the way with subtle changes in force. Furthermore, they remarked that they were concentrating too much on that cursor, trying to push it towards the target’s center instead of stopping when the green highlight matched the desired target.

Taking all data from Study 1 into account, we found BC as the fastest and most accurate reachability technique. However, BC’s Success was not significantly better than FR’s, and FR caused the least and almost no device motion that was significantly different from all tested techniques, and it allowed users to select all targets without leaving the thumb’s comfortable region. Yet, users found controlling the force the main challenge. Encouraged by the findings from Cortzen et al. [15], we wondered whether users could improve their performance by training. Therefore we conducted a second user study with trained users testing the two most promising and preferred techniques: BC and FR.

6 GUIDELINES

Based on what we have learned from Study 1 (S1) and Study 2 (S2), we give recommendations for the tested techniques for different criteria and contexts:

When selection speed is ultimately critical, e.g., in games, DT is the technique of choice, despite grip changes that users have to perform for far targets (S1).

When direct touch input is important and targets are ≥ 69 px. OM is a good compromise to also satisfy reachability for one-handed mobile touchscreen use.

For a good speed-accuracy trade-off, BC is recommended (S1, S2). When the 3f has many targets located at the center, yet beyond the thumb’s reach, BC is fast and accurate and causes moderate device motion, likely without the need for a grip change. However, for targets located near and at the upper edge, such as menus or the iOS back button, users tend to tilt the device towards them to reach the target and often re-grasp the device to ease this.

For such targets, and, in general, when device and grip stability are important, FR is recommended (S1, S2). For example, apps that use the smartphone camera, such as apps for taking photos and videos, scanning documents, or augmented reality games, it is important to keep the camera focussed at a static viewpoint. Also, when ergonomics are key, FR is most beneficial, since the user’s thumb stays within its comfort region and rotates naturally around the CMC joint.
Discussion

Overall, DT was fastest but achieved low Success for small targets, matching previous findings (e.g., from [2, 3]). DT also caused the strongest device motion, since especially at extremely far positions, like the upper and lower left corner, users had to change their grip, for which some participants almost accidentally dropped the phone, matching findings from [13]. This was why users preferred DT the least. Surprisingly, OM caused the second highest device motion.

Zone for OM was highest, reaching almost 2,000 ms. This could be due to the double tap gestures, which, unlike BC and FR, does not contribute to the target selection process, and due to the 200 ms animation time that helps the user understand how the UI is transformed. Both take additional time: Success for OM was also low, especially for small targets (Fig. 4 bottom: 81.1%); Shrinkin them makes them too hard to hit precisely due to the thumb's fat finger problem [40].

A solution like AngleFoe ([8] that captures the finger's orientation, could help gaining touch precision.

We had the highest task completion time and lowest Success. Participants found MS more demanding than other techniques, because it required good planning upfront. To reach the top left corner, e.g., the user must place her thumb at least above the center of the screen, so that dragging it to the lower right corner will move the cursor far enough in the opposite direction. When the thumb is placed improperly, it is not possible to correct this within a trial, which explains MS' low Success for both, large and small targets. Using the model from [2], we found that for most of our users, the center of the 5.5" screen is not reachable without uncomfortably stretching the thumb or changing the device grip.

Among the reachability techniques, BC was fastest, which was also independent from target Size. While users ranked BC best, they remarked that targets at the top were more difficult to reach due to swinging long distances upwards. To ease this, users tilted the device (Fig. 4) towards the thumb, for which they sometimes even had to reposition their grip.

Although non-significant, Success for FR was 4% lower than for BC. The Quick Release selection mechanism used in FR could be one explanation, since a lift-off based on force is a less definite event than a lift-off only considering touch information, as used for BC. Unfamiliarity with force control could also explain the lower Success. Crenen et al. [11] showed that users' performance for force control significantly increased with training. Our participants reported that selecting close and far targets with FR was fundamentally easier compared to other targets because then they only had to apply minimum or maximum force. Some developed a strategy: Independent of the target location at the border, they first applied maximum force to create a ray with the cursor positioned at its end and then, while maintaining this force, moved the ray onto the target.

Interestingly, some participants remarked that they found FR's continuous cursor rather a disadvantage and that the discrete green indicator highlighting the preslected target was sufficient. Despite the 3C Filter [5], these users were worried when the cursor would move along the ray with subtle changes in force. Furthermore, they remarked that they were concentrating too much on this cursor, trying to push it towards the target's center instead of stopping when the green highlight matched the desired target.

Taking all data from Study 1 into account, we found BC as the fastest and most accurate reachability technique. However, BC's Success was not significantly better than FR's, and FR caused the least and almost no device motion that was significantly different from all tested techniques, and it allowed users to select all targets without leaving the thumb's comfortable region. Yet, users found controlling these force the main challenge. Encouraged by the findings from Crenen et al. [11], we wondered whether users could improve their performance by training. We therefore conducted a second user study with trained users testing the two most promising and preferred techniques: BC and FR.

6 GUIDELINES

Based on what we have learned from Study 1 (S1) and Study 2 (S2), we give recommendations for the tested techniques for different criteria and contexts:

When selection speed is ultimately critical, e.g., in games, DT is the technique of choice, despite grip changes that users have to perform for far targets (S1).

When direct touch input is important and targets are 2-69 pt, OM is a good compromise to also satisfy reachability for one-handed mobile touchscreen use.

For a good speed-accuracy trade-off, BC is recommended (S1, S2). When the UI has many targets located at the center, yet beyond the thumb's reach, BC is fast and accurate and causes moderate device motion, likely without the need for a grip change. However, for targets located near and at the upper edge, such as menus or the iOS back button, users tend to lift the device towards them to reach the target and often re-grasp the device to ease this. For such targets, and, in general, when device and grip stability are important, FR is recommended (S1, S2). For example, apps that use the smartphone camera, such as apps for taking photos and videos, scanning documents, or augmented reality games, it is important to keep the camera focused at a static viewpoint. Also, when ergonomics are key, FR is most beneficial, since the user's thumb stays within its comfort region and rotates naturally around the CMC joint.
8 SUMMARY AND CONCLUSION

In summary, we presented ForceRay (FR), an interaction technique that extends thumb reach via force input to enable selection of out-of-reach targets on mobile touchscreens with a steady device grip. To select a target that cannot be reached easily with the thumb, the user applies a force touch to cast a virtual ray in the direction of the target. By increasing her force, she moves a cursor along the ray until reaching the target, then lifts her thumb quickly to confirm the selection. We conducted two user studies to validate FR. Study 1 tested FR against existing reachability solutions. Among all, FR significantly caused the least device motion, removing users’ concerns about device drops. Yet, FR was 195 ms slower than the fastest reachability technique, BezelCursor (BC). Study 2 showed that an hour of training sped up both BC and FR, and that both are equally fast for targets at the screen border.

ACKNOWLEDGMENTS

This work was funded by the Europäischer Fonds für regionale Entwicklung (EFRE) and the German B-IT Foundation.

REFERENCES


7 LIMITATIONS AND FUTURE WORK

FR performance is affected by handedness: For targets located at the right screen edge, right-handed users cannot exploit target selection via maximum possible force, except for the upper right corner target, since the ray will cross all of these targets. This is why we did not consider target 24 a border target in the analysis. For left-handed users, targets at the left screen edge are affected. Furthermore, due to the polar coordinate system of the ray casting in FR, distant targets require more precise movement the smaller such targets are. In addition, FR sacrifices the benefit of direct manipulation (like BC) that both DT and OM share. Moreover, FR interferes with existing force input techniques within the thumb’s comfortable region, such as force-touching an app icon in iOS. Using a different trigger, e.g., a subtle force touch that quickly drops it without lifting the finger off the screen (‘Force Pulse’ in [11]), could mitigate this problem. Also, FR is only designed for targets responding to taps, but could be extended as follows: To discriminate taps from gestures, the user could dwell for 1 s on the distant target, and then reset the force by dropping it, yet without lifting the thumb off the screen: if the target responds to force, the user just presses. Swipes, instead, could be issued by Force Pulses, and scroll gestures could be issued by rolling the thumb left and right as in ForcePicker [11]. Apart from testing this, we also plan a long-term study in which users control existing apps with FR while being in motion: On the one hand, walking negatively affects force control [47], on the other hand, FR’s grip stability could then finally lead to a better performance compared to other reachability techniques. Finally, we would like to investigate whether a hybrid of BC and FR could combine the speed and stability benefits of both techniques: BC would be used as trigger and for generally aiming at out-of-reach targets. However, if a target is not comfortably reachable by BC anymore or causes significant device motion, the user could continue to extend the cursor using FR.
icon in iOS. Using a different trigger, e.g., a subtle force touch that quickly drops it without lifting the finger off the screen (‘Force Pulse’ in [11]), could mitigate this problem. Also, FR is only designed for targets responding to taps, but could be extended as follows: To discriminate taps from gestures, the user could dwell for 1 s on the distant target, and then reset the force by dropping it, yet without lifting the thumb off the screen. If the target responds to force, the user just presses. Swipes, instead, could be issued by Force Pulses, and scroll gestures could be issued by rolling the thumb left and right as in ForcePicker [11]. Apart from testing this, we also plan a long-term study in which users control existing apps with FR while being in motion: On the one hand, walking negatively affects force control [47]; on the other hand, FR’s grasp stability could then finally lead to a better performance compared to other reachability techniques. Finally, we would like to investigate whether a hybrid of BC and FR could combine the speed and stability benefits of both techniques: BC would be used as trigger and for generally aiming at out-of-reach targets. However, if a target is not comfortably reachable by BC anymore or causes significant device motion, the user could continue to extend the cursor using FR.

8 SUMMARY AND CONCLUSION

In summary, we presented ForkPick (FR), an interaction technique that extends thumb reach via force input to enable selection of out-of-reach targets on mobile touchscreens with a steady device grip. To select a target that cannot be reached easily with the thumb, the user applies a force touch to cast a virtual ray in the direction of the target. By increasing her force, she moves a cursor along the ray until reaching the target, then lifts her thumb quickly to confirm the selection. We conducted two user studies to validate FR. Study 1 tested FR against existing reachability solutions. Among all, FR significantly caused the least device motion, removing users’ concerns about device drops. Yet, FR was 195 ms slower than the fastest reachability technique, BezelCursor (BC). Study 2 showed that an hour of training sped up both BC and FR, and that both are equally fast for targets at the screen border.

ACKNOWLEDGMENTS

This work was funded by the Europäischer Fonds für regionale Entwicklung (EFRE) and the German B-IT Foundation.

REFERENCES


7 LIMITATIONS AND FUTURE WORK

FR performance is affected by handedness: For targets located at the right screen edge, right-handed users cannot exploit target selection via maximum possible force, except for the upper right corner target, since the ray will cross all of these targets. This is why we did not consider target 24 a border target in the analysis. For left-handed users, targets at the left screen edge are affected. Furthermore, due to the polar coordinate system of the ray casting in FR, distant targets require more precise movement the smaller such targets are. In addition, FR sacrifices the benefit of direct manipulation (like BC) that both DT and OM share. Moreover, FR interferes with existing force input techniques within the thumb’s comfortable region, such as force-touching an app.