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

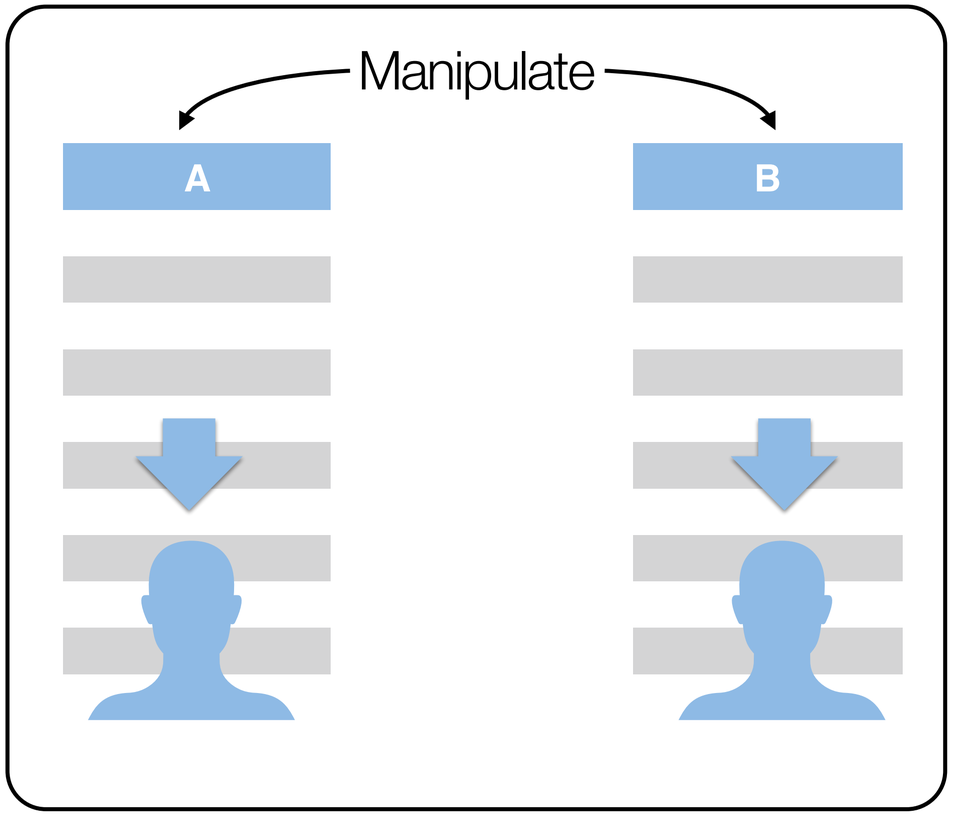
https://hci.rwth-aachen.de/cthci

RWTH AACHEN UNIVERSITY

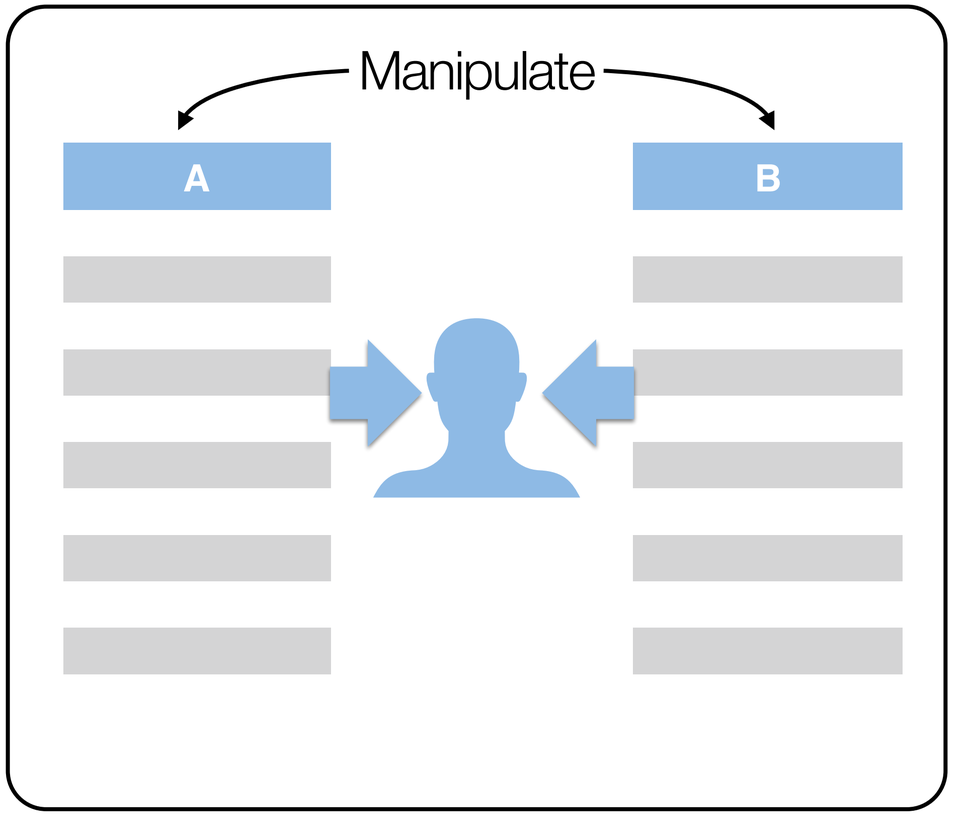
Experimental Variables: Measurement Scales

- **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



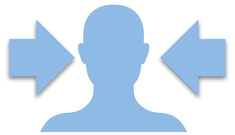
Within-groups design

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



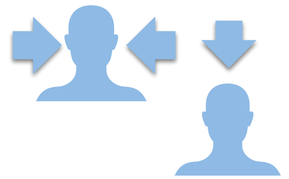
- Problem of within-groups design
- Behavior may be influenced by experience that occurred earlier in the sequence
- **Carryover effects:** changes caused by the lingering aftereffects of an earlier treatment condition
  - E.g., testing the first condition causes users' fingers to hurt, degrading their 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

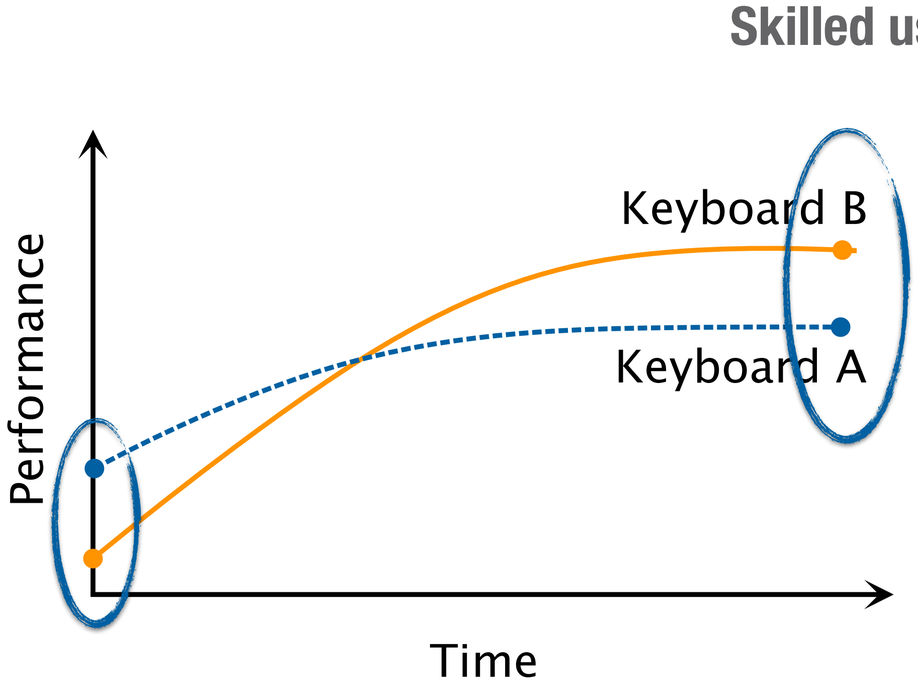
- Tries to negate order effects
- Ideally: Use every possible order of treatments with an equal number of individual participants
- Compromise: Latin Square
  - Each condition appears at each ordinal position
  - Each condition precedes and follows each condition once
  - Example: six treatments A, B, C, D, E, F

1	A	B	F	C	E	D
2	B	C	A	D	F	E
3	C	D	B	E	A	F
4	D	E	C	F	B	A
5	E	F	D	A	C	B
6	F	A	E	B	D	C

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

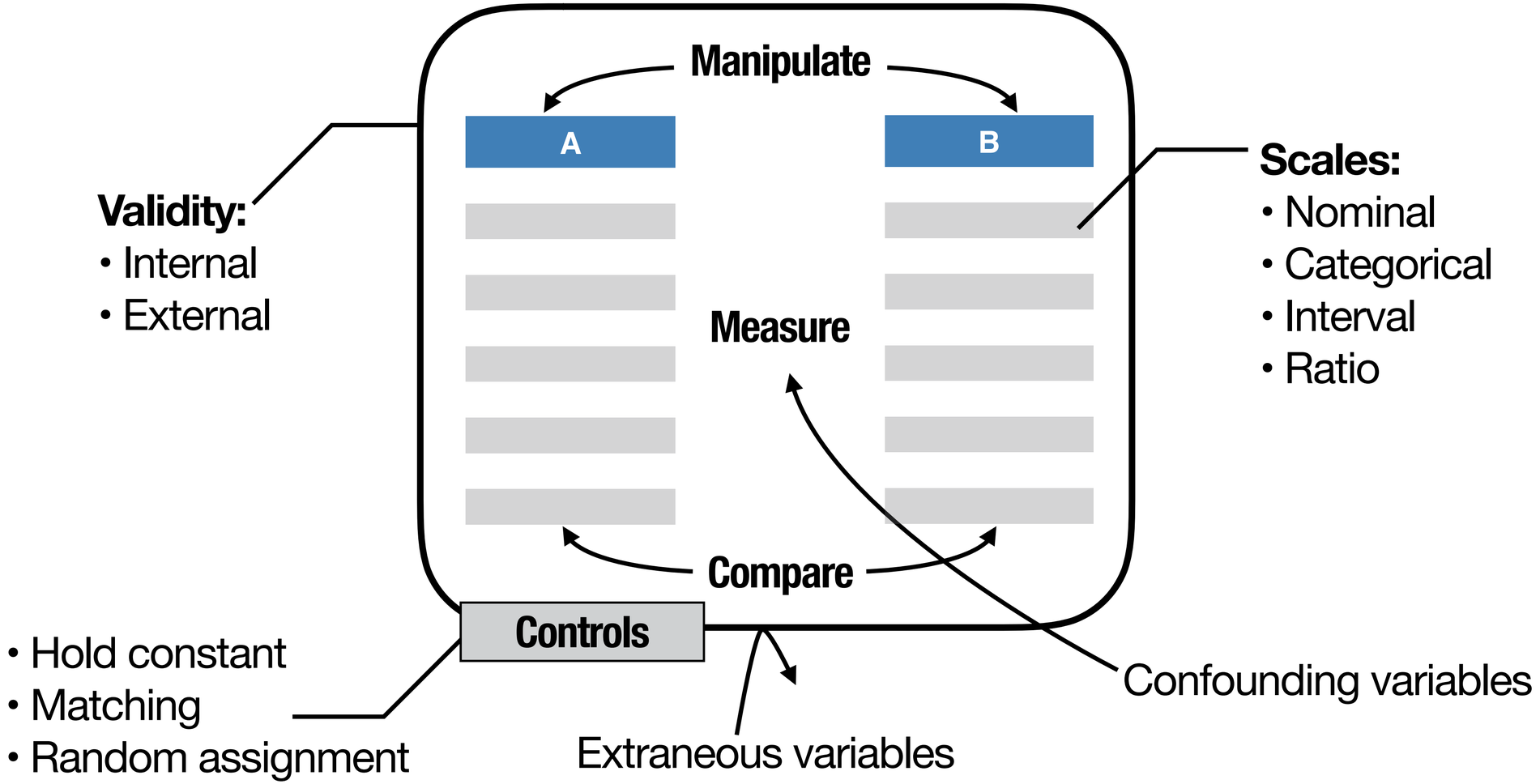
- You have designed a new keyboard layout, and you want to know how good it is
- Strategy: compare it to existing techniques

- Describe one reason to choose a
  - **Within-groups design**
  - **Between-groups design**

Experimental Design In-class Exercise #1

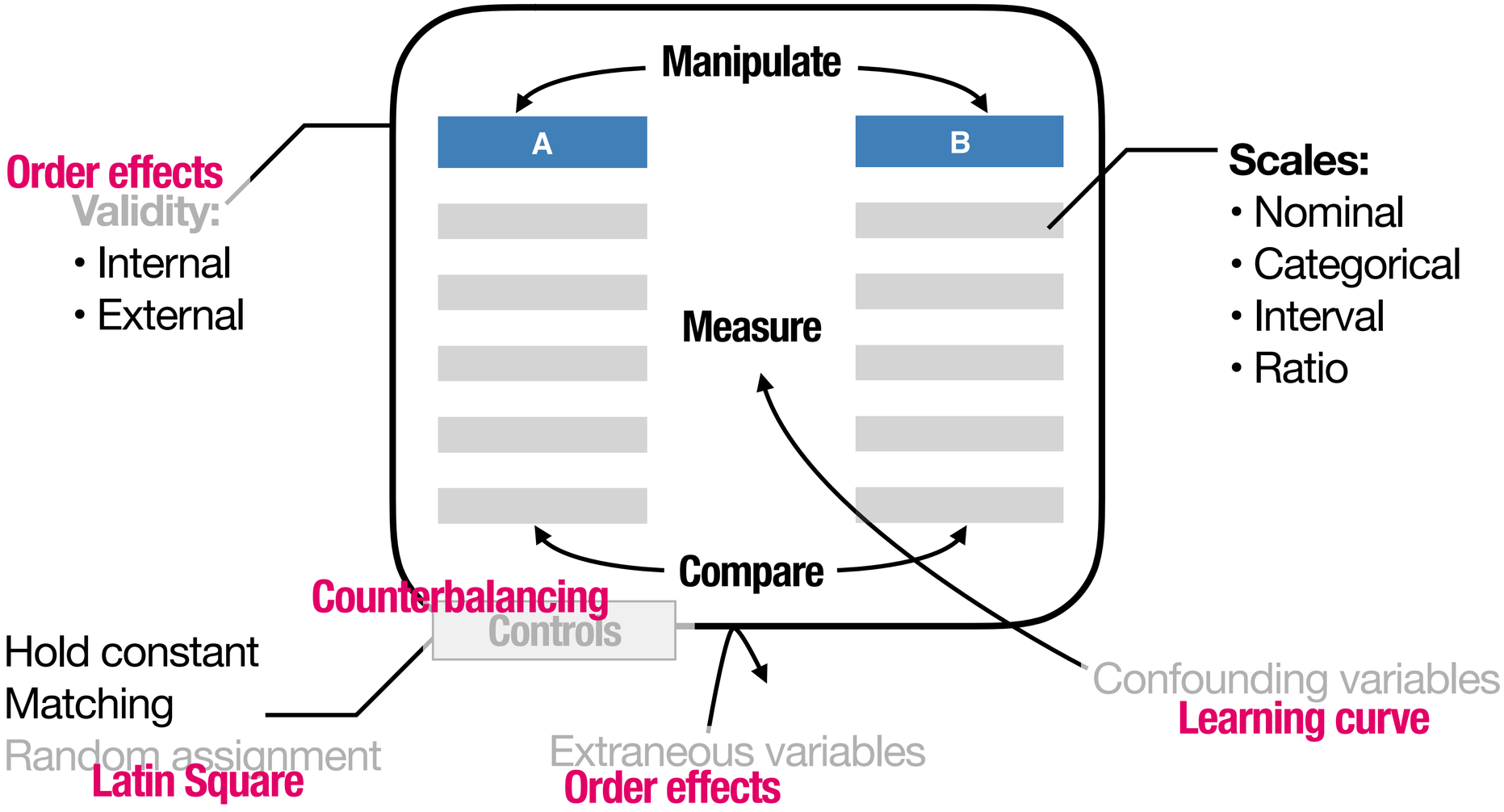
- Usually preferred: **within-groups design**
  - Minimizes confounding effects from the behavioral differences between participants
- Sometimes, we need a **between-groups** design
  - E.g., when testing whether a keyboard favors users with right-handedness over those with left-handedness
  - When there are interferences between conditions, e.g., different keyboard layouts on the same hardware

In-class Exercise #2: Experimental Study Basics



Learning curve
Counterbalancing
Latin Square
Order effects

In-class Exercise #2: Experimental Study Basics



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

ForceRay: Extending Thumb Reach via Force Input Stabilizes Device Grip for Mobile Touch Input

Christian Corsten, Marcel Lahaye, Jan Borchers, and Simon Voelker
RWTH Aachen University
Aachen, Germany
lastname@cs.rwth-aachen.de

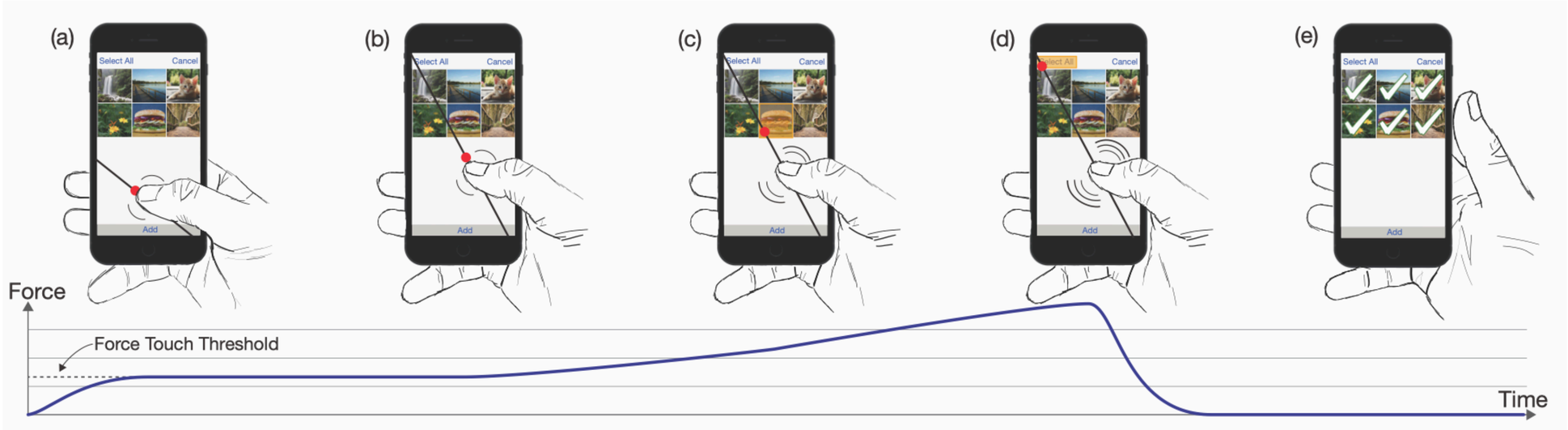


Figure 1: The user wants to tap the button in the top left corner of her smartphone screen to select all images at once. Reaching this button, however, is hardly possible without changing the device grip, which leads to increased device motion and thus is likely to result in a device drop. The *ForceRay* 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.

Heading & Figure 1

Contribution?

Benefit?

Originality?

Validity?

Novelty?

Format?

RWTH AACHEN UNIVERSITY

ForceRay: Extending Thumb Reach via Force Input Stabilizes Device Grip for Mobile Touch Input

Christian Corsten, Marcel Lahaye, Jan Borchers, and Simon Voelker
RWTH Aachen University
Aachen, Germany
lastname@cs.rwth-aachen.de

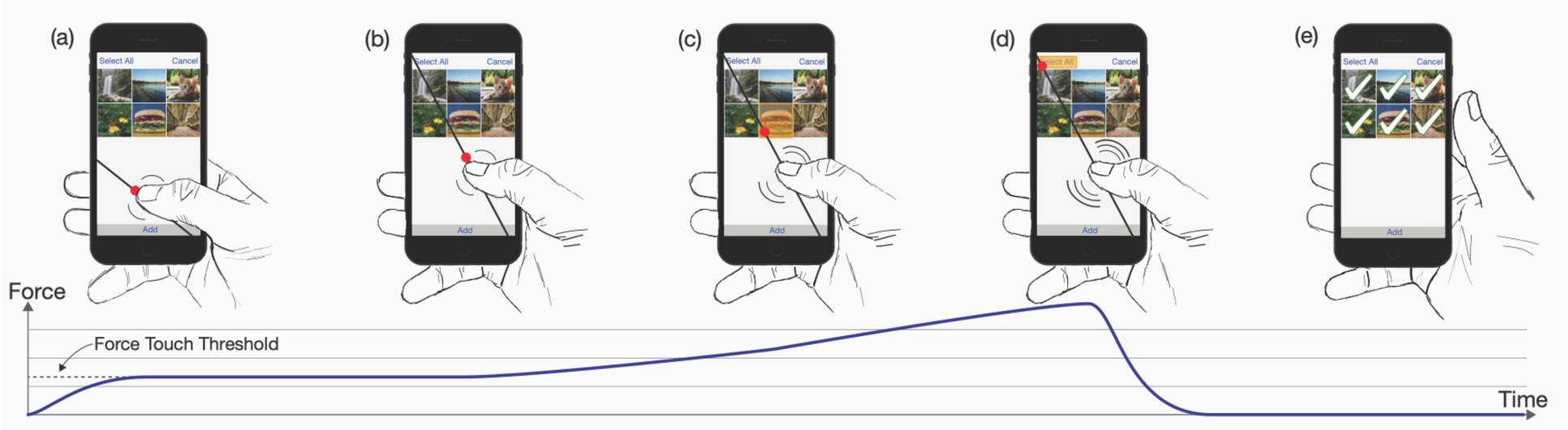


Figure 1: The user wants to tap the button in the top left corner of her smartphone screen to select all images at once. Reaching this button, however, is hardly possible without changing the device grip, which leads to increased device motion and thus is likely to result in a device drop. The *ForceRay* 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.

Heading & Figure 1

Contribution

Benefit

RWTH AACHEN UNIVERSITY

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.

CHI 2019, May 4–9, 2019, Glasgow, Scotland UK

ACM ISBN 978-1-4503-5970-2/19/05…$15.00
https://doi.org/10.1145/3290605.3300442

CCS CONCEPTS

• **Human-centered computing** → **Gestural input**;

KEYWORDS

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

ACM Reference Format:
Christian Corsten, Marcel Lahaye, Jan Borchers, and Simon Voelker. 2019. ForceRay: Extending Thumb Reach via Force Input Stabilizes Device Grip for Mobile Touch Input. In *CHI Conference on Human Factors in Computing Systems Proceedings (CHI 2019), May 4–9, 2019, Glasgow, Scotland UK*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3290605.3300442

Abstract

Contribution?

Benefit?

Originality?

Validity?

Novelty?

Format?

RWTH AACHEN UNIVERSITY

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.

CHI 2019, May 4–9, 2019, Glasgow, Scotland UK

ACM ISBN 978-1-4503-5970-2/19/05…$15.00
https://doi.org/10.1145/3290605.3300442

CCS CONCEPTS

• **Human-centered computing** → **Gestural input**;

KEYWORDS

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

ACM Reference Format:
Christian Corsten, Marcel Lahaye, Jan Borchers, and Simon Voelker. 2019. ForceRay: Extending Thumb Reach via Force Input Stabilizes Device Grip for Mobile Touch Input. In *CHI Conference on Human Factors in Computing Systems Proceedings (CHI 2019), May 4–9, 2019, Glasgow, Scotland UK*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3290605.3300442

Abstract

Contribution

Novelty

RWTH AACHEN UNIVERSITY

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 *BezelCursor* (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.

Introduction

Contribution?

Benefit?

Originality?

Validity?

Novelty?

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 *BezelCursor* (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.

Introduction

Contribution / Originality

Benefit

Novelty

Format

2 RELATED WORK

FR 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 this in mind, Bergstrom 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 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 screen likewise when tilting the device. Sliding Screen [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 touchpad at the back of the device (BoD). Löchtefeld et al. [31] detect which hand unlocked the device to shift the UI towards that hand. Eardley et al. [13, 14] present several adaptive UI concepts, e.g., a keyboard that shifts to the user's thumb when the device is tilted sideways.

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 makes reading the screen difficult at certain angles [42].

Proxy Region Techniques. TapTap [39] magnifies a part of the screen around the thumb's touch location in a pop-up view. ThumbSpace [23, 24] uses 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 occlude a part of the screen. Hasan et al. [19] proposed the mid-air space between thumb and touchscreen as proxy region. Löchtefeld et al. [30] used a BoD touchpad to reach upper targets with the index finger from behind. Such proxy region extends the thumb's reach by 15% [50]. However, these techniques require hardware modifications.

Cursor Techniques. TiltCursor [7] lets users drag an accelerated cursor with the thumb and is triggered by tilting. BezelCursor [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 triggered by a double tap and—similar to BezelCursor [29]—extends the thumb with a cursor whose offset increases with dragging speed. Extendible Cursor [26] is similar but steers the cursor opposite to where the user drags her thumb. MagStick [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 stepping through UI elements by tiny swipe gestures, which, however, is tedious and time-consuming. CornerSpace and BezelSpace [51] also require only little thumb movement to reach targets at screen corners and edges and are triggered by a bezel swipe: BezelSpace lets the user control a cursor like an extended finger tip though a proxy region. CornerSpace initially places a remote cursor at corners to access them quickly and allows for selecting nearby targets through dragging with automatic snapping. However, these techniques make selection of other targets less accurate. Dual-Surface Input [49] uses a BoD touchpad with an absolute mapping to select screen targets by tapping with the index finger at the BoD. Yet, like other BoD solutions, hardware modifications are needed.

Force Input Techniques

Force adds a third dimension to touch. Davidson et al. [12] and Qiu et al. [35] use force to move objects along the z-dimension on the touchscreen. Apple's 3D Touch [22] uses force as a modifier to pop up context menus. Boring et al. [3] use the thumb's contact size as simulated force to toggle panning and zooming on a smartphone. Goel et al. [17] discriminate three levels of simulated force from gyroscope readings when the user is pressing against the touchscreen while the smartphone vibration motor is pulsed. Brewster et al. [4] use a force modifier to conveniently type uppercase letters, and ForceBoard [52] maps force intensity to character selection to type on a smartphone keyboard while looking at a distant screen. ForceEdge [1] maps force to the velocity of scrolling though long lists. For smaller lists and menus, force is typically mapped directly to selectable values. This way, users can control six to ten items on a 3–10 N pressure range with visual feedback in a stationary context [6, 32, 33, 43, 48]. Other examples control such menus from the bezel [32, 41, 42, 46] or BoD [9] with the fingers grasping the device. Hence, force input keeps hand and fingers in place to enable a stable device grip.

None of these solutions exploited force input benefits to solve reachability issued for one-handed smartphone use.

3 THE FORCERAY INTERACTION TECHNIQUE

ForceRay (FR) follows a simple sequence of small interaction steps: When the user places her thumb on the touchscreen without applying significant force, FR is inactive. Once the user starts applying force and crosses a *resting threshold* (this is known as a *force touch*), FR is activated as long as the user maintains force above that threshold. Upon activation, a ray is displayed that virtually extends the user's thumb to the opposing screen edge (Fig. 1 a). The ray's placement is determined by the touch location and a reference point that represents the thumb's carpometacarpal (CMC) joint location. This is the joint around which the user rotates her thumb to steer the ray, and is approximately located in the lower right (left) screen corner for right (left) handed users.

Usually, the user will initiate the ray at a point that makes the ray 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 (Fig. 1 b–c). With the ray comes a cursor that the user controls via force input to highlight any target that intersects with the ray. The more force is applied, the farther the cursor moves upwards on the ray towards the opposite screen edge (Fig. 1 c–d); reducing the force makes the cursor move downwards again. To make this mapping consistent, i.e., have the same change in force always result in the same cursor travel distance, FR maps the available force range between force threshold and maximum sensible 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 *positional control*, is often applied to force input (e.g., [48]) and allows for instant corrections to over- and undershoots by simply decreasing or increasing the force.

Mathematically, FR's cursor position is a polar coordinate (r, Φ): r is the distance to the CMC reference point, and Φ is the angle between the lower screen edge and the ray.

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 in which the cursor will reach them, 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 DynaSpot [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 force input (e.g., [4, 10, 36, 48]). If, however, the user wants to cancel without selecting a target, she reduces force until it is below the force touch threshold and then lifts her thumb.

FR has three key benefits: **Scalability.** It scales to arbitrary mobile screen sizes since all targets can be reached from within the functional area of the thumb, given that the force sensor is strong enough such that the user can select each target on the ray. **Efficiency.** Independent from the screen size, FR makes selection of targets located at corners and edges efficient: applying maximum possible force immediately highlights the most distant target crossed by the ray. The iOS back button located in the upper left screen corner is an apt example illustrating this benefit: it is notoriously hard to reach with one-handed input otherwise, but becomes very quick and easy to select with FR. **Visibility.** With FR, the user does not occlude content of interest with her thumb, since only the thumb's functional area is partially occluded.

Related Work

Contribution?

Benefit?

Originality?

Validity?

Novelty?

Format?

2 RELATED WORK

FR 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 this in mind, Bergstrom 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 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 screen likewise when tilting the device. Sliding Screen [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 touchpad at the back of the device (BoD). Löchtefeld et al. [31] detect which hand unlocked the device to shift the UI towards that hand. Eardley et al. [13, 14] present several adaptive UI concepts, e.g., a keyboard that shifts to the user's thumb when the device is tilted sideways.

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 makes reading the screen difficult at certain angles [42].

Proxy Region Techniques. TapTap [39] magnifies a part of the screen around the thumb's touch location in a pop-up view. ThumbSpace [23, 24] uses 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 occlude a part of the screen. Hasan et al. [19] proposed the mid-air space between thumb and touchscreen as proxy region. Löchtefeld et al. [30] used a BoD touchpad to reach upper targets with the index finger from behind. Such proxy region extends the thumb's reach by 15% [50]. However, these techniques require hardware modifications.

Cursor Techniques. TiltCursor [7] lets users drag an accelerated cursor with the thumb and is triggered by tilting. BezelCursor [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 triggered by a double tap and—similar to BezelCursor [29]—extends the thumb with a cursor whose offset increases with dragging speed. Extendible Cursor [26] is similar but steers the cursor opposite to where the user drags her thumb. MagStick [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 stepping through UI elements by tiny swipe gestures, which, however, is tedious and time-consuming. CornerSpace and BezelSpace [51] also require only little thumb movement to reach targets at screen corners and edges and are triggered by a bezel swipe: BezelSpace lets the user control a cursor like an extended finger tip though a proxy region. CornerSpace initially places a remote cursor at corners to access them quickly and allows for selecting nearby targets through dragging with automatic snapping. However, these techniques make selection of other targets less accurate. Dual-Surface Input [49] uses a BoD touchpad with an absolute mapping to select screen targets by tapping with the index finger at the BoD. Yet, like other BoD solutions, hardware modifications are needed.

Force Input Techniques

Force adds a third dimension to touch. Davidson et al. [12] and Qiu et al. [35] use force to move objects along the z-dimension on the touchscreen. Apple's 3D Touch [22] uses force as a modifier to pop up context menus. Boring et al. [3] use the thumb's contact size as simulated force to toggle panning and zooming on a smartphone. Goel et al. [17] discriminate three levels of simulated force from gyroscope readings when the user is pressing against the touchscreen while the smartphone vibration motor is pulsed. Brewster et al. [4] use a force modifier to conveniently type uppercase letters, and ForceBoard [52] maps force intensity to character selection to type on a smartphone keyboard while looking at a distant screen. ForceEdge [1] maps force to the velocity of scrolling though long lists. For smaller lists and menus, force is typically mapped directly to selectable values. This way, users can control six to ten items on a 3–10 N pressure range with visual feedback in a stationary context [6, 32, 33, 43, 48]. Other examples control such menus from the bezel [32, 41, 42, 46] or BoD [9] with the fingers grasping the device. Hence, force input keeps hand and fingers in place to enable a stable device grip.

None of these solutions exploited force input benefits to solve reachability issued for one-handed smartphone use.

3 THE FORCERAY INTERACTION TECHNIQUE

ForceRay (FR) follows a simple sequence of small interaction steps: When the user places her thumb on the touchscreen without applying significant force, FR is inactive. Once the user starts applying force and crosses a *resting threshold* (this is known as a *force touch*), FR is activated as long as the user maintains force above that threshold. Upon activation, a ray is displayed that virtually extends the user's thumb to the opposing screen edge (Fig. 1 a). The ray's placement is determined by the touch location and a reference point that represents the thumb's carpometacarpal (CMC) joint location. This is the joint around which the user rotates her thumb to steer the ray, and is approximately located in the lower right (left) screen corner for right (left) handed users

Usually, the user will initiate the ray at a point that makes the ray 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 (Fig. 1 b–c). With the ray comes a cursor that the user controls via force input to highlight any target that intersects with the ray. The more force is applied, the farther the cursor moves upwards on the ray towards the opposite screen edge (Fig. 1 c–d); reducing the force makes the cursor move downwards again. To make this mapping consistent, i.e., have the same change in force always result in the same cursor travel distance, FR maps the available force range between force threshold and maximum sensible 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 *positional control*, is often applied to force input (e.g., [48]) and allows for instant corrections to over- and undershoots by simply decreasing or increasing the force.

Mathematically, FR's cursor position is a polar coordinate (r, Φ): r is the distance to the CMC reference point, and Φ is the angle between the lower screen edge and the ray.

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 in which the cursor will reach them, 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 DynaSpot [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 force input (e.g., [4, 10, 36, 48]). If, however, the user wants to cancel without selecting a target, she reduces force until it is below the force touch threshold and then lifts her thumb.

FR has three key benefits: **Scalability.** It scales to arbitrary mobile screen sizes since all targets can be reached from within the functional area of the thumb, given that the force sensor is strong enough such that the user can select each target on the ray. **Efficiency.** Independent from the screen size, FR makes selection of targets located at corners and edges efficient: applying maximum possible force immediately highlights the most distant target crossed by the ray. The iOS back button located in the upper left screen corner is an apt example illustrating this benefit: it is notoriously hard to reach with one-handed input otherwise, but becomes very quick and easy to select with FR. **Visibility.** With FR, the user does not occlude content of interest with her thumb, since only the thumb's functional area is partially occluded.

Related Work

Contribution / Originality

Benefit

Novelty

Format

4 STUDY 1: REACHABILITY TECHNIQUES

To understand how FR compares to the state of the art, we conducted a user study with 15 participants (21–33 years, $M = 25.73$, $SD = 3.31$; two female; all right-handed; thumb length: $M = 75.87$ mm, $SD = 7.12$ mm). They were all smartphone users (screen size: $M = 5.36$", $SD = .46$"). We compared FR to One-Handed Mode (OM, similar to Samsung's), MagStick (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 by 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 touchscreen using each of these techniques while holding the device in their right hand in portrait orientation.

Apparatus, Techniques, and Task

We used an iPhone 6S Plus to present 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 unitless force values between 0 and $\frac{480}{72} \approx 6.67$ in steps of $\frac{1}{72}$, with force sensitivity set to "firm". Values around 1.0 should be interpreted as an ordinary touch; higher values as intentional force input. 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 736×414 pt. For iPhone 6S/7/8 Plus, 1 pt ≈ .16 mm.

Techniques. **One-Handed Mode (OM)** mimics Samsung's reachability technique: by default, it downscales 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 triple tap on the Home button, we chose iOS' double tap gesture for activation: Swiping could have been confounding with BC's trigger, and pilot tests revealed that users found triple taps confusing to perform. Since the iOS SDK does not notify about touch 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 265 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 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 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 end of the line has a circular cursor that expands exponentially up to 7.3 mm depending on how fast the user swipes her thumb. This area cursor is equivalent to DynaSpot [8], and shrinks co-exponentially when swiping acceleration falls below 2 $\frac{mm}{s}$. When a target is intersected by the cursor, it is highlighted. When multiple targets are crossed, the target with the smallest distance from its 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 touch threshold to 1.33 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 [10] using the default parameters. Pilot testing, however, revealed that these did not fit all users, leading to increased selection errors. We followed [10] to determine individual timing parameters by calibration trials upfront ($M = 48$–224 ms before complete thumb lift-off, $SD = 32$–48 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 in the lower right corner would not allow users to comfortable move the ray to the right edge. For mapping the force input to the cursor position (C_{pos}) on the ray, 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 decreased from 50% to 25% from force touch threshold to maximum sensible force: $C_{pos}(F_{rel}) = (\frac{2}{5}\ln(F_{rel} + \frac{3}{5}) + .0893) \cdot s + \delta$, where F_{rel} is the relative force ranging from force touch threshold to maximum sensible force, s denotes a scaling factor for the screen size, and δ represents the distance from the ray origin to where the cursor should start on the ray. For our setup, we set $s = 1{,}852$ pt and $\delta = 248$ pt minus the dynamic distance from the thumb's touch location to the CMC reference point. Furthermore, to reduce ray and cursor jitter, we filtered touch location and force using the 1€ Filter [5] ($\alpha = .1$).

Targets. Targets were arranged on an invisible 6×10 grid (Fig. 2) across an area of 414×730 pt; each cell measured 69×73 pt. The bottom 414×6 pt of the screen was excluded to obtain cell sizes 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 touch input for targets that are out of reach, distractors close to the thumb (bottom right 4×5 cells) were not selectable. Participants were informed about this and a slight brighter background 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 500 ms. For OM, the UI was automatically shifted back right before showing the next trial.

Variables. **Independent Variables** were Technique (DT, OM, MS, BC, and FR), Target, that split our twelve targets (Fig. 2) into two groups: targets 1, 4, 6, 19, 37, and 55 located at the Border of the screen vs. the remaining six targets rather located at the Center, and their corresponding Size (small: 30×30 pt (4.8×4.8 mm) and large: 60×60 pt (9.6×9.6 mm)). Size represented typical UI widget sizes, like the height of a button (30 pt) or an app icon (60 pt) on the iOS Home Screen.

We recorded 5 Technique × 12 targets × 2 Size × 2 repetitions = 240 trials per user. Technique was counter-balanced using a Latin Square. Targets were randomized. Size was also randomized, but we ensured that each user started testing a new 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, the next Technique was presented. Including test trials, each user did 280 trials in approximately 45 minutes.

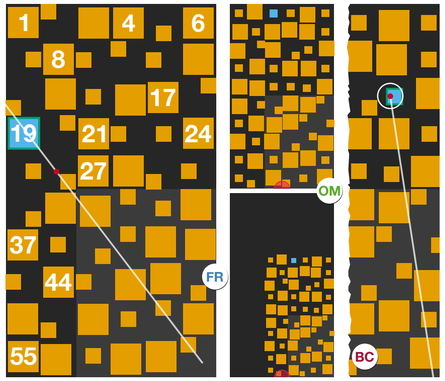


Figure 2: Targets (numbered) and distractors arranged on an invisible 6×10 grid. FR: layout for the large target Size condition. The user is aiming for the blue target that is crossed by the ForceRay. Currently, the target with the green border is selected since it is the next target on the ray beyond the cursor. OM (miniaturized): Target layout for the small target Size condition. Top: Full layout before double-tapping the virtual home button (pink). Bottom: UI 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 (DynaSpot area cursor). Targets in the lower right area (brighter background) were not selectable, since here, targets are directly accessible by the thumb.

Dependent Variables were trial completion *Time* [ms], users' *Success* [0,1], i.e., 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 [°] around x-, y-, and z-axis at 60 Hz. After each Technique, users were asked how much they agreed to (i) that they had to regularly change 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

Methods

Contribution?

Benefit?

Originality?

Validity?

Novelty?

Format?

force readings provided by the force-sensitive touchscreen. According to Apple's documentation [21], the force sensor API delivers unitless force values between 0 and $\frac{480}{72} \approx 6.67$ in steps of $\frac{1}{72}$, with force sensitivity set to "firm". Values around 1.0 should be interpreted as an ordinary touch; higher values as intentional force input. 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 736×414 pt. For iPhone 6S/7/8 Plus, 1 pt ≈ .16 mm.

Techniques. **One-Handed Mode (OM)** mimics Samsung's reachability technique: by default, it downscales 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 triple tap on the Home button, we chose iOS' double tap gesture for activation: Swiping could have been confounding with BC's trigger, and pilot tests revealed that users found triple taps confusing to perform. Since the iOS SDK does not notify about touch 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 265 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 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 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 end of the line has a circular cursor that expands exponentially up to 7.3 mm depending on how fast the user swipes her thumb. This area cursor is equivalent to DynaSpot [8], and shrinks co-exponentially when swiping acceleration falls below 2 $\frac{mm}{s}$. When a target is intersected by the cursor, it is highlighted. When multiple targets are crossed, the target with the smallest distance from its 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 touch threshold to 1.33 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 [10] using the default parameters. Pilot testing, however, revealed that these did not fit all users, leading to increased selection errors. We followed [10] to determine individual timing parameters by calibration trials upfront (M = 48–224 ms before complete thumb lift-off, SD = 32–48 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 in the lower right corner would not allow users to comfortable move the ray to the right edge. For mapping the force input to the cursor position (C_{pos}) on the ray, 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 decreased from 50% to 25% from force touch threshold to maximum sensible force: $C_{pos}(F_{rel}) = (\frac{2}{5}\ln(F_{rel} + \frac{3}{5}) + .0893) \cdot s + \delta$, where F_{rel} is the relative force ranging from force touch threshold to maximum sensible force, s denotes a scaling factor for the screen size, and δ represents the distance from the ray origin to where the cursor should start on the ray. For our setup, we set s = 1,852 pt and δ = 248 pt minus the dynamic distance from the thumb's touch location to the CMC reference point. Furthermore, to reduce ray and cursor jitter, we filtered touch location and force using the 1€ Filter [5] (α = .1).

Targets. Targets were arranged on an invisible 6×10 grid (Fig. 2) across an area of 414×730 pt; each cell measured 69×73 pt. The bottom 414×6 pt of the screen was excluded to obtain cell sizes 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 touch input for targets that are out of reach, distractors close to the thumb (bottom right 4×5 cells) were not selectable. Participants were informed about this and a slight brighter background 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 500 ms. For OM, the UI was automatically shifted back right before showing the next trial.

Variables. **Independent Variables** were Technique (DT, OM, MS, BC, and FR), Target, that split our twelve targets (Fig. 2) into two groups: targets 1, 4, 6, 19, 37, and 55 located at the Border of the screen vs. the remaining six targets rather located at the Center, and their corresponding Size (small: 30×30 pt (4.8×4.8 mm) and large: 60×60 pt (9.6×9.6 mm)). Size represented typical UI widget sizes, like the height of a button (30 pt) or an app icon (60 pt) on the iOS Home Screen.

We recorded 5 Technique × 12 targets × 2 Size × 2 repetitions = 240 trials per user. Technique was counter-balanced using a Latin Square. Targets were randomized. Size was also randomized, but we ensured that each user started testing a new 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, the next Technique was presented. Including test trials, each user did 280 trials in approximately 45 minutes.

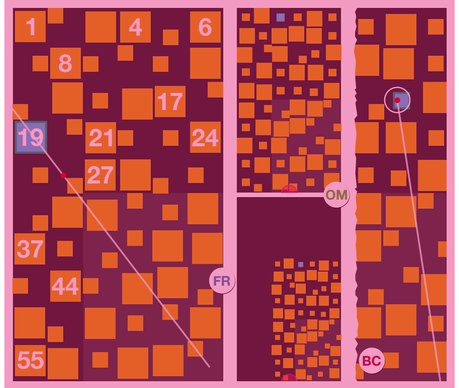


Figure 2: Targets (numbered) and distractors arranged on an invisible 6×10 grid. FR: layout for the large target Size condition. The user is aiming for the blue target that is crossed by the ForceRay. Currently, the target with the green border is selected since it is the next target on the ray beyond the cursor. OM (miniaturized): Target layout for the small target Size condition. Top: Full layout before double-tapping the virtual home button (pink). Bottom: UI 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 (DynaSpot area cursor). Targets in the lower right area (brighter background) were not selectable, since here, targets are directly accessible by the thumb.

Dependent Variables were trial completion *Time* [ms], users' *Success* [0,1], i.e., 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 [°] around x-, y-, and z-axis at 60 Hz. After each Technique, users were asked how much they agreed to (i) that they had to regularly change 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

4 STUDY 1: 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 = 25.73, SD = 3.31; two female; all right-handed; thumb length: M = 75.87 mm, SD = 7.12 mm). They were all smartphone users (screen size: M = 5.36", SD = .46"). We compared FR to One-Handed Mode (OM, similar to Samsung's), MagStick (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 by 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 touchscreen using each of these techniques while holding the device in their right hand in portrait orientation.

Apparatus, Techniques, and Task

We used an iPhone 6S Plus to present the task to our users and to capture data. For our implementation of FR, we used the

Methods

Originality

Validity

Novelty

Format

Results

Since we were interested in how users' performance is different depending on the TECHNIQUE 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 [13, 15] by summing up the absolute angles 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.

TECHNIQUE had a significant main effect on ***Time*** ($F_{4,3565}$ = 348.95, 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 ≈ 324 ms longer. FR was the third fastest TECHNIQUE, yet less than ≈ 200 ms slower than BC. OM and MS were close to 2,000 ms. There was also a TECHNIQUE × TARGET interaction effect ($F_{4,3565}$ = 39.02, p <.001). Fig. 4 (top) list the Tukey HSD post hoc test results. For each TECHNIQUE except FR, users needed significantly more time to select Border targets compared to Center targets. For FR, on the contrary, this was reversed.

TECHNIQUE had a significant main effect on ***Success*** ($Q(4)$ = 81.02, p <.001, Fig. 3, right). Post hoc tests revealed that *Success* for BC was significantly higher compared to all other TECHNIQUES except FR. Furthermore, FR and DT yielded significantly higher *Success* than OM and MS. There was also a TECHNIQUE × SIZE interaction effect ($Q(9)$ = 201.30, p <.001). Fig. 4 (bottom) shows the results from the post hoc tests. For small targets, BC yielded significantly higher *Success* than DT, OM, and MS, and FR had significantly higher *Success* than OM and DT. For large targets, MS had significantly lower *Success* than all other TECHNIQUES. Only for DT, there was a significant difference for *Success* comparing both SIZES: Small targets had 11% lower *Success* than large targets.

TECHNIQUE had a significant main effect on ***Rotation*** around the *x*- ($F_{4,3566}$ = 995.81, p <.001), *y*- ($F_{4,3566}$ = 778.61, p <.001), and *z-axis* ($F_{4,3566}$ = 563.33, p <.001). For the *y*- and *z-axis*, Tukey HSD post hoc tests revealed that all TECHNIQUES were significantly different from each other (Fig. 5). For the *x-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 (all: p <.001). In summary, for each angle, FR always caused the fewest device movement. There were also TECHNIQUE × TARGET interaction effects for the *x-axis* ($F_{4,3566}$ = 4.30, p = .002) and for the *y-axis* ($F_{4,3566}$ = 4.52, p = .001). Fig. 6 lists the results from the Tukey HSD post hoc tests. In general, for each TECHNIQUE, acquiring targets at the Border resulted in more device movement around both, the *x*- and *y-axis*, compared to Center targets.

Fig. 7 shows the ***Gesture Footprint*** generated by each TECHNIQUE. FR caused the smallest and most coherent footprint, and touches stayed within the thumb's comfortable reach, following natural rotation around the CMC joint.

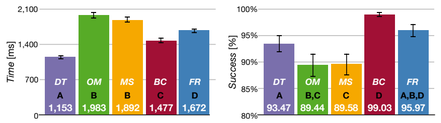


Figure 3: Study 1: *Time* [ms] (left) and *Success* [%] (right) by TECHNIQUE. For each variable, pairs of levels that do not share a letter are significantly different (*Time*: all p <.001, *Success*: all p <.05). Whiskers denote 95% CI.

	DT M	DT CI	OM M	OM CI	MS M	MS CI	BC M	BC CI	FR M	FR CI
SIZE										
Small	1,222	±65 A	2,074	±98 A	2,049	±93 A	1,574	±96 A	1,790	±61 A
Large	1,084	±39 A	1,892	±72 A	1,735	±72 A	1,381	±56 A	1,554	±53 A
TARGET										
Border	1,266	±67 A	2,148	±94 B	2,070	±91 B	1,552	±96 C	1,520	±44 C
Center	1,039	±33 A	1,818	±75 B	1,714	±73 B	1,403	±57 C	1,824	±66 B

	DT M	DT CI	OM M	OM CI	MS M	MS CI	BC M	BC CI	FR M	FR CI
SIZE										
Small	88.06	±2.95 A	81.11	±3.71 B,D	88.89	±2.84 A,D,E	98.89	±.68 A,C	94.72	±1.87 C,E
Large	99.17	±.55 A	97.78	±1.09 B	90.28	±2.65 B	99.17	±.55 B	97.22	±1.26 B
TARGET										
Border	93.06	±2.20 A	86.11	±3.19 B	86.67	±3.13 B	96.89	±.68 A	98.33	±.90 A
Center	93.89	±2.04 A,B	92.78	±2.25 A	92.5	±2.29 A	99.17	±.55 B	93.61	±2.09 A,B

Figure 4: Study 1: *Time* [ms] (top) and *Success* [%] (bottom) by TECHNIQUE × SIZE and TECHNIQUE × TARGET. Yellow cells denote significant differences within TECHNIQUE (*Time* and *Success*: all p <.001). Pairs of levels that do not share a letter are significantly different across TECHNIQUE (*Time*: all p <.001, *Success*: all p <.05). CI denotes 95% CI.

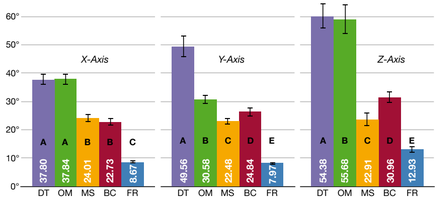


Figure 5: Study 1: *Rotation* [°] for the *x*- (left), *y*- (middle), and *z-axis* (right) by TECHNIQUE. For each variable, pairs of levels that do not share a letter are significantly different (all p <.001). Whiskers denote 95% CI. FR caused almost no device movement.

	DT M	DT CI	OM M	OM CI	MS M	MS CI	BC M	BC CI	FR M	FR CI
SIZE										
Small	39.30	±2.82 A	38.52	±2.83 A	24.40	±1.87 A	23.64	±2.07 A	8.93	±0.61 A
Large	36.29	±2.25 A	37.16	±2.63 A	23.63	±2.22 A	21.83	±1.48 A	8.41	±0.58 A
TARGET										
Border	44.12	±2.94 A	41.73	±2.90 A	28.36	±2.34 B	25.54	±2.02 B	9.47	±0.61 C
Center	31.48	±1.88 A	33.95	±2.49 A	19.67	±1.60 B	19.92	±1.49 B	7.87	±0.57 C

X-Axis

	DT M	DT CI	OM M	OM CI	MS M	MS CI	BC M	BC CI	FR M	FR CI
SIZE										
Small	49.20	±5.22 A	30.81	±2.16 A	23.44	±1.74 A	27.62	±2.30 A	8.39	±0.56 A
Large	49.56	±5.8 A	30.58	±2.05 A	22.48	±1.77 A	24.84	±1.76 A	7.97	±0.64 A
TARGET										
Border	57.84	±6.40 A	33.80	±2.46 B	26.60	±1.90 C	28.61	±2.32 C	8.74	±0.57 E
Center	40.92	±4.29 A	27.58	±1.60 B	19.33	±1.50 C	23.85	±1.71 D	7.62	±0.63 E

Y-Axis

	DT M	DT CI	OM M	OM CI	MS M	MS CI	BC M	BC CI	FR M	FR CI
SIZE										
Small	65.80	±6.65	62.22	±8.18	24.36	±3.70	31.90	±2.29	13.08	±1.34
Large	54.38	±5.63	55.68	±6.56	22.91	±2.91	30.96	±2.97	12.93	±1.62
TARGET										
Border	69.11	±6.92	65.19	±8.05	27.44	±3.07	34.12	±2.60	14.41	±1.65
Center	51.07	±5.20	52.71	±6.67	19.83	±3.53	28.74	±2.68	11.60	±1.28

Z-Axis

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

Fig. 8 shows the mean and 95% CI for the questionnaire data. ***Grip change*** had a significant effect on TECHNIQUE ($\chi^2(4)$ = 36.19, p <.001). Regarding post hoc tests, users stated significantly more grip changes for DT compared to all other TECHNIQUES (all: p <.05) The same trend was found for ***grip stability*** ($\chi^2(4)$ = 36.89, p <.001, post hoc tests: all: p < .05). For ***ease of use*** ($\chi^2(4)$ = 18.27, p = .001), users found BC significantly easier to apply than MS (p = .001). TECHNIQUE had also a significant effect on users' ***ranking*** ($\chi^2(4)$ = 21.55, p <.001). Overall, participants preferred BC most, followed by FR, OM, MS, and DT, with BC being significantly preferred over all other TECHNIQUES (all: p <.05) except FR.

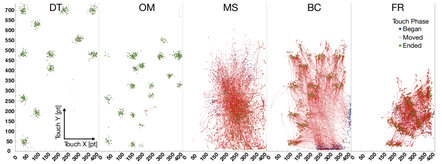


Figure 7: Study 1: *Gesture Footprint* by TECHNIQUE. Blue dots mark where users started placing their thumb, red dots represent dragging, and green dots indicate where users lifted their thumb. FR had the smallest coherent footprint and followed ergonomic thumb movement.

	"I had to change my grip regularly." M	CI		*"I maintained a stable grip while selecting."* M	CI		*"The technique was easy to apply."* M	CI		Rank 1: Highest, 5: Lowest M	CI	
DT	6.20	±.87	A	2.13	±.65	A	5.20	±.92	A,B	4.00	±.66	B
OM	3.27	±1.08	B	4.53	±1.00	B	5.33	±.83	A,B	3.13	±.65	B
MS	2.27	±.80	B	5.60	±.78	B	4.53	±.91	A	3.47	±.63	B
BC	2.73	±.87	B	4.80	±.92	B	6.47	±.29	B	1.47	±.47	A
FR	1.33	±.27	B	6.40	±.28	B	5.33	±.80	A,B	2.93	±.79	A,B

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

Results

Contribution?

Benefit?

Originality?

Validity?

Novelty?

Format?

Results

Since we were interested in how users' performance is different depending on the TECHNIQUE 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 [13, 15] by summing up the absolute angles 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.

TECHNIQUE had a significant main effect on ***Time*** ($F_{4,3565}$ = 348.95, 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 ≈ 324 ms longer. FR was the third fastest TECHNIQUE, yet less than ≈ 200 ms slower than BC. OM and MS were close to 2,000 ms. There was also a TECHNIQUE × TARGET interaction effect ($F_{4,3565}$ = 39.02, p <.001). Fig. 4 (top) list the Tukey HSD post hoc test results. For each TECHNIQUE except FR, users needed significantly more time to select Border targets compared to Center targets. For FR, on the contrary, this was reversed.

TECHNIQUE had a significant main effect on ***Success*** ($Q(4)$ = 81.02, p <.001, Fig. 3, right). Post hoc tests revealed that *Success* for BC was significantly higher compared to all other TECHNIQUES except FR. Furthermore, FR and DT yielded significantly higher *Success* than OM and MS. There was also a TECHNIQUE × SIZE interaction effect ($Q(9)$ = 201.30, p <.001). Fig. 4 (bottom) shows the results from the post hoc tests. For small targets, BC yielded significantly higher *Success* than DT, OM, and MS, and FR had significantly higher *Success* than OM and DT. For large targets, MS had significantly lower *Success* than all other TECHNIQUES. Only for DT, there was a significant difference for *Success* comparing both SIZES: Small targets had 11% lower *Success* than large targets.

TECHNIQUE had a significant main effect on ***Rotation*** around the *x*- ($F_{4,3566}$ = 995.81, p <.001), *y*- ($F_{4,3566}$ = 778.61, p <.001), and *z-axis* ($F_{4,3566}$ = 563.33, p <.001). For the *y*- and *z-axis*, Tukey HSD post hoc tests revealed that all TECHNIQUES were significantly different from each other (Fig. 5). For the *x-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 (all: p <.001). In summary, for each angle, FR always caused the fewest device movement. There were also TECHNIQUE × TARGET interaction effects for the *x-axis* ($F_{4,3566}$ = 4.30, p = .002) and for the *y-axis* ($F_{4,3566}$ = 4.52, p = .001). Fig. 6 lists the results from the Tukey HSD post hoc tests. In general, for each TECHNIQUE, acquiring targets at the Border resulted in more device movement around both, the *x*- and *y-axis*, compared to Center targets.

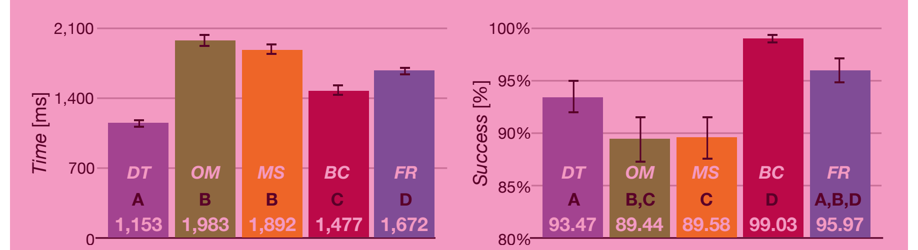


Figure 3: Study 1: *Time* [ms] (left) and *Success* [%] (right) by TECHNIQUE. For each variable, pairs of levels that do not share a letter are significantly different (*Time*: all p <.001, *Success*: all p <.05). Whiskers denote 95% CI.

	DT *M*	DT CI	DT	OM *M*	OM CI	OM	MS *M*	MS CI	MS	BC *M*	BC CI	BC	FR *M*	FR CI	FR
SIZE															
Small	1,222	±65	A	2,074	±98	A	2,049	±93	A	1,574	±96	A	1,790	±61	A
Large	1,084	±39	A	1,892	±72	A	1,735	±72	A	1,381	±56	A	1,554	±53	A
TARGET															
Border	1,266	±67	A	2,148	±94	B	2,070	±91	B	1,552	±96	C	1,520	±44	C
Center	1,039	±33	A	1,818	±75	B	1,714	±73	B	1,403	±57	C	1,824	±66	B

	DT *M*	DT CI	DT	OM *M*	OM CI	OM	MS *M*	MS CI	MS	BC *M*	BC CI	BC	FR *M*	FR CI	FR
SIZE															
Small	88.06	±2.95	A	81.11	±3.71	B,D	88.89	±2.84	A,D,E	98.89	±.68	A,C	94.72	±1.87	C,E
Large	99.17	±.55	A	97.78	±1.09	B	90.28	±2.65	B	99.17	±.55	B	97.22	±1.26	B
TARGET															
Border	93.06	±2.20	A	86.11	±3.19	B	86.67	±3.13	B	98.89	±.68	A	98.33	±.90	A
Center	93.89	±2.04	A,B	92.78	±2.25	A	92.5	±2.29	A	99.17	±.55	B	93.61	±2.09	A,B

Figure 4: Study 1: *Time* [ms] (top) and *Success* [%] (bottom) by TECHNIQUE × SIZE and TECHNIQUE × TARGET. Yellow cells denote significant differences within TECHNIQUE (*Time* and *Success*: all p <.001). Pairs of levels that do not share a letter are significantly different across TECHNIQUE (*Time*: all p <.001, *Success*: all p <.05). CI denotes 95% CI.

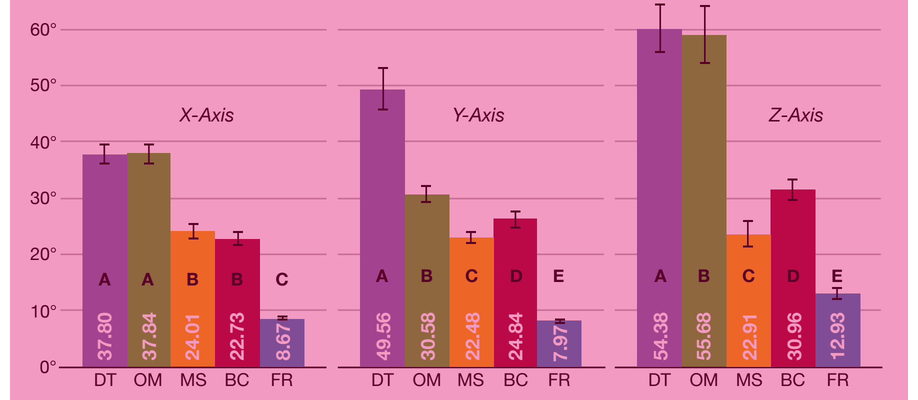


Figure 5: Study 1: *Rotation* [°] for the *x*- (left), *y*- (middle), and *z-axis* (right) by TECHNIQUE. For each variable, pairs of levels that do not share a letter are significantly different (all p <.001). Whiskers denote 95% CI. FR caused almost no device movement.

Fig. 7 shows the ***Gesture Footprint*** generated by each TECHNIQUE. FR caused the smallest and most coherent footprint, and touches stayed within the thumb's comfortable reach, following natural rotation around the CMC joint.

X-Axis

	DT *M*	DT CI	DT	OM *M*	OM CI	OM	MS *M*	MS CI	MS	BC *M*	BC CI	BC	FR *M*	FR CI	FR
SIZE															
Small	39.30	±2.82	A	38.52	±2.83	A	24.40	±1.87	A	23.64	±2.07	A	8.93	±0.61	A
Large	36.29	±2.25	A	37.16	±2.63	A	23.63	±2.22	A	21.83	±1.48	A	8.41	±0.58	A
TARGET															
Border	44.12	±2.94	A	41.73	±2.90	A	28.36	±2.34	B	25.54	±2.02	B	9.47	±0.61	C
Center	31.48	±1.88	A	33.95	±2.49	A	19.67	±1.60	B	19.92	±1.49	B	7.87	±0.57	C

Y-Axis

	DT *M*	DT CI	DT	OM *M*	OM CI	OM	MS *M*	MS CI	MS	BC *M*	BC CI	BC	FR *M*	FR CI	FR
SIZE															
Small	49.20	±5.22	A	30.81	±2.16	A	23.44	±1.74	A	27.62	±2.30	A	8.39	±0.56	A
Large	49.56	±5.8	A	30.58	±2.05	A	22.48	±1.77	A	24.84	±1.76	A	7.97	±0.64	A
TARGET															
Border	57.84	±6.40	A	33.80	±2.46	B	26.60	±1.90	C,	28.61	±2.32	C	8.74	±0.57	E
Center	40.92	±4.29	A	27.58	±1.60	B	19.33	±1.50	C	23.85	±1.71	D	7.62	±0.63	E

Z-Axis

	DT *M*	DT CI	OM *M*	OM CI	MS *M*	MS CI	BC *M*	BC CI	FR *M*	FR CI
SIZE										
Small	65.80	±6.65	62.22	±8.18	24.36	±3.70	31.90	±2.29	13.08	±1.34
Large	54.38	±5.63	55.68	±6.56	22.91	±2.91	30.96	±2.97	12.93	±1.62
TARGET										
Border	69.11	±6.92	65.19	±8.05	27.44	±3.07	34.12	±2.60	14.41	±1.65
Center	51.07	±5.20	52.71	±6.67	19.83	±3.53	28.74	±2.68	11.60	±1.28

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

Fig. 8 shows the mean and 95% CI for the questionnaire data. ***Grip change*** had a significant effect on TECHNIQUE ($\chi^2(4)$ = 36.19, p <.001). Regarding post hoc tests, users stated significantly more grip changes for DT compared to all other TECHNIQUES (all: p <.05) The same trend was found for ***grip stability*** ($\chi^2(4)$ = 36.89, p <.001, post hoc tests: all: p < .05). For ***ease of use*** ($\chi^2(4)$ = 18.27, p = .001), users found BC significantly easier to apply than MS (p = .001). TECHNIQUE had also a significant effect on users' ***ranking*** ($\chi^2(4)$ = 21.55, p <.001). Overall, participants preferred BC most, followed by FR, OM, MS, and DT, with BC being significantly preferred over all other TECHNIQUES (all: p <.05) except FR.

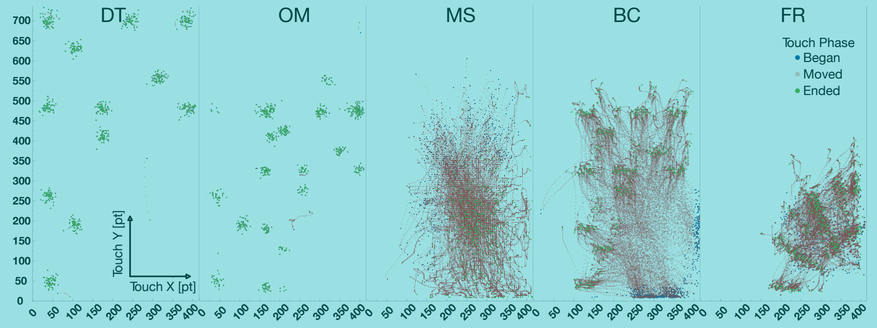


Figure 7: Study 1: *Gesture Footprint* by TECHNIQUE. Blue dots mark where users started placing their thumb, red dots represent dragging, and green dots indicate where users lifted their thumb. FR had the smallest coherent footprint and followed ergonomic thumb movement.

	"I had to change my grip regularly." *M*	CI		"I maintained a stable grip while selecting." *M*	CI		"The technique was easy to apply." *M*	CI		Rank 1: Highest, 5: Lowest *M*	CI	
DT	6.20	±.87	A	2.13	±.65	A	5.20	±.92	A,B	4.00	±.66	B
OM	3.27	±1.08	B	4.53	±1.00	B	5.33	±.83	A,B	3.13	±.65	B
MS	2.27	±.80	B	5.60	±.78	B	4.53	±.91	A	3.47	±.63	B
BC	2.73	±.87	B	4.80	±.92	B	6.47	±.29	B	1.47	±.47	A
FR	1.33	±.27	B	6.40	±.28	B	5.33	±.80	A,B	2.93	±.79	A,B

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

Results

Contribution / Originality

Benefit

Validity

Format

Discussion

Overall, DT was fastest but achieved low *Success* for small targets, matching previous findings (e.g., from [23, 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 265 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: 81.11%): Shrinking them makes them too hard to hit precisely due to the thumb's fat finger problem [40]. A solution like AnglePose [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' 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. 6) 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*: Corsten 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 1€ 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 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 their force the main challenge. Encouraged by the findings from Corsten 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 $\geq$ 60 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 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 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.

Discussion

Contribution?

Benefit?

Originality?

Validity?

Novelty?

Clarity?

Discussion

Overall, DT was fastest but achieved low *Success* for small targets, matching previous findings (e.g., from [23, 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 265 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: 81.11%): Shrinking them makes them too hard to hit precisely due to the thumb's fat finger problem [40]. A solution like AnglePose [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' 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. 6) 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*: Corsten 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 1€ 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 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 their force the main challenge. Encouraged by the findings from Corsten 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 $\geq$ 60 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 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 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.

Discussion

Contribution / Originality

Benefit

Validity

Novelty

Format

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 longterm 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 investigte 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 *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.

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

ACKNOWLEDGMENTS

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

REFERENCES

[1] Axel Antoine, Sylvain Malacria, and Géry Casiez. 2017. ForceEdge: Controlling Autoscroll on Both Desktop and Mobile Computers Using the Force. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 3281–3292. https://doi.org/10.1145/3025453.3025605

[2] Joanna Bergstrom-Lehtovirta and Antti Oulasvirta. 2014. Modeling the Functional Area of the Thumb on Mobile Touchscreen Surfaces. In

Limitations, Future & Summmary

Contribution?

Benefit?

Originality?

Validity?

Novelty?

Format?

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 longterm 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 investigte 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 *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.

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

ACKNOWLEDGMENTS

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

REFERENCES

[1] Axel Antoine, Sylvain Malacria, and Géry Casiez. 2017. ForceEdge: Controlling Autoscroll on Both Desktop and Mobile Computers Using the Force. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 3281–3292. https://doi.org/10.1145/3025453.3025605

[2] Joanna Bergstrom-Lehtovirta and Antti Oulasvirta. 2014. Modeling the Functional Area of the Thumb on Mobile Touchscreen Surfaces. In

Limitations, Future & Summary

Contribution / Originality

Benefit

Validity

Novelty

Format