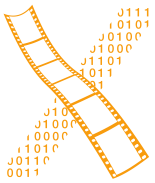

Chair for Computer Science 10 (Media Computing and Human-Computer Interaction)

RWTH AACHEN UNIVERSITY

Proximity Dependent Interfaces

Bachelor's Thesis
submitted to the
Media Computing Group
Prof. Dr. Jan Borchers
Computer Science Department
RWTH Aachen University

by
Krzysztof Adam Kostrzewa

Thesis advisor:
Prof. Dr. Jan Borchers

Second examiner:
Prof. Dr. Ulrik Schroeder

Registration date: 2.06.2021
Submission date: 4.10.2021

Zentrales Prüfungsamt/Central Examination Office

Eidesstattliche Versicherung
Statutory Declaration in Lieu of an Oath

Name, Vorname/Last Name, First Name

Matrikelnummer (freiwillige Angabe)
Matriculation No. (optional)

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Arbeit/Bachelorarbeit/ Masterarbeit* mit dem Titel
I hereby declare in lieu of an oath that I have completed the present paper/Bachelor thesis/Master thesis* entitled

selbstständig und ohne unzulässige fremde Hilfe (insbes. akademisches Ghostwriting) erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.
independently and without illegitimate assistance from third parties (such as academic ghostwriters). I have used no other than the specified sources and aids. In case that the thesis is additionally submitted in an electronic format, I declare that the written and electronic versions are fully identical. The thesis has not been submitted to any examination body in this, or similar, form.

Ort, Datum/City, Date

Unterschrift/Signature

*Nichtzutreffendes bitte streichen

*Please delete as appropriate

Belehrung:
Official Notification:

§ 156 StGB: Falsche Versicherung an Eides Statt
Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.
Para. 156 StGB (German Criminal Code): False Statutory Declarations
Whoever before a public authority competent to administer statutory declarations falsely makes such a declaration or falsely testifies while referring to such a declaration shall be liable to imprisonment not exceeding three years or a fine.
§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt
(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.
(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.
Para. 161 StGB (German Criminal Code): False Statutory Declarations Due to Negligence
(1) If a person commits one of the offences listed in sections 154 through 156 negligently the penalty shall be imprisonment not exceeding one year or a fine.
(2) The offender shall be exempt from liability if he or she corrects their false testimony in time. The provisions of section 158 (2) and (3) shall apply accordingly.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:
I have read and understood the above official notification:

Ort, Datum/City, Date

Unterschrift/Signature

Contents

List of Figures

Abstract

In this work we set out to analyse the landscape of *Proximity Dependent Interfaces* to find how it can be used on smaller devices. We conducted an extensive literature research and found that most small to medium-sized devices use simpler interfaces compared to their bigger counterparts. To find out what interactions are viable on smaller devices we conducted two studies to find out how much space in front of the device is really usable and what kind of distance tracking performs best. Based on our results we propose guidelines for using distance based interface on small to medium devices. We also propose three example interfaces (for laptops and smartphones) using our guidelines.

Überblick

In dieser Arbeit haben wir uns vorgenommen, das Themenbereich von *Proximity Dependent Interfaces* zu analysieren, um herauszufinden, wie es auf kleineren Geräten anwendet werden kann. Wir haben eine umfangreiche Literaturrecherche durchgeführt und festgestellt, dass die meisten kleinen bis mittelgroßen Geräte im Vergleich zu ihren größeren Gegenstücken einfachere Interfaces verwenden. Um herauszufinden, welche Interaktionen auf kleineren Geräten möglich sind, haben wir zwei Studien durchgeführt, um herauszufinden, wie viel Platz vor dem Gerät wirklich nutzbar ist und welche Art von Entfernungsverfolgung am besten funktioniert. Basierend auf unseren Ergebnissen schlagen wir Richtlinien für die Verwendung von entfernungsbasierten Schnittstellen auf kleinen bis mittleren Geräten vor. Außerdem schlagen wir drei Beispiel Interfaces (für Laptops und Smartphones) anhand unserer Richtlinien vor.

Chapter 1

Introduction

Humans naturally use the spatial relation to the object they interact with to make the interaction easier. For example, if we are trying to read very small text on a piece of paper, we might bring it closer to our eyes to make it bigger, thus easier to read. But what if instead of a piece of paper, it was a computer that could react to how close we are and would zoom in on the content of the screen for us.

This kind of spatial awareness is the first of the two parts of *Proximity Dependent Interfaces* and has its roots in the theory of *Proxemics* proposed by anthropologist Edward T. Hall in his book *The Hidden Dimension* [13]. *Proxemics* focus on how people (often unconsciously) use proxemic cues (e.g., distance, orientation) as a part of communication. One of the most influential ideas proposed in this book was the correlation between physical and social distance. Hall divided this distance into four discrete zones: intimate ($0 - 50cm$), personal ($0.5 - 1m$), social ($1 - 4m$), and public ($> 4m$) (depicted in Figure 1.1).

In the field of *Human Computer Interactions* the idea of devices knowing about their location, surroundings and reacting to them was nothing new. That is why *Proxemics* is used mostly for Ubicomp like systems [3, 12, 22, 15]. Of course, Halls theory was focused on interhuman interactions, so it had to be operationalized to make it understandable (measurable) for computers. That is what Greenberg

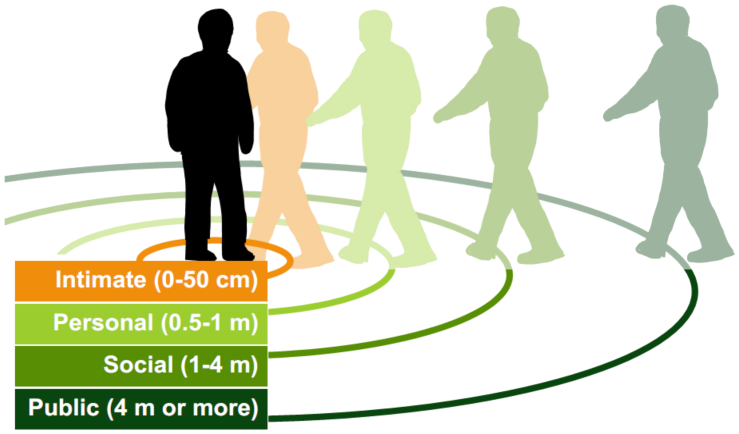

Figure 1.1: Halls' Proxemic Zones. Adapted from Marquardt and Greenberg [2011]

et al. [2011] did in *Proxemic Interactions: The New Ubicomp?*. They defined five essential dimensions for determining basic proxemic relationships (see Figure 1.2), described below:

Distance The distance between entities. It can be either continuous (in cm/m/etc.) or discrete (like Hall's proxemic zones).

Orientation The direction of entity's focus in space. For example, in the case of the human, it would be the direction of their face or gaze. It requires that the entity is orientable (has a "front").

Identity The ability to distinguish different entities from each other. It has various levels of complexity from entity A is not entity B, to entity A is this concrete person with concrete attributes.

Movement Changes of distance and orientation of entities over time.

Location Physical space in which the interaction takes place.

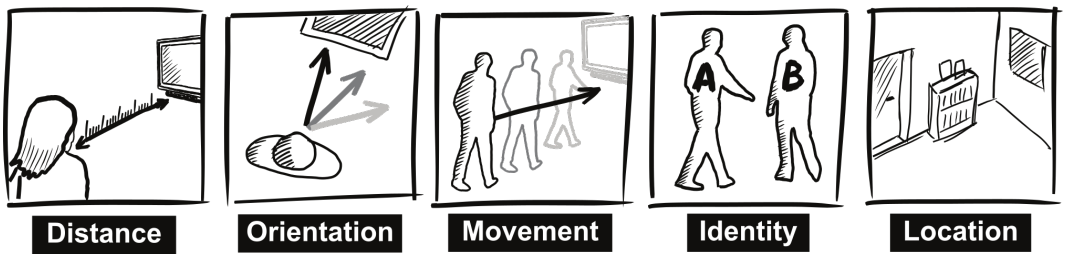

Figure 1.2: The five essential dimensions of proxemics. Adapted from Greenberg et al. [2011]

The second part of *Proximity Dependent Interfaces* are interfaces. Going back to the example from the first paragraph, simply making the text bigger when we get closer is already an improvement. As showed by Harrison and Dey [2008] in *Lean and Zoom*. They evaluated very similar interaction, where the screen of a laptop was zoomed in when the user got closer to the screen. This improved comfort of use and the posture of the user. But simple zooming in only mimics real word interaction. Computer interfaces are much more flexible and thus are not limited by the same physical limitations non-interactive objects are. This means there are many potential interactions, some of which we will examine in Related Work.

1.1 My Focus

As mentioned above, most *proxemic interaction* research focuses on ubicomp like environments (more about it in Related Work). But the same principles can be used to enrich the interaction with medium and small sized devices.

Because those devices have less space for sensors and some of the interactions that make sense in Ubicomp-like systems do not work here, we have decided to focus our work on distance as an input modality only. As shown by Chen et al. [2014] a camera is enough to reliably track it and most, if not all, consumer devices now have a front-facing camera. This makes this kind of interaction applicable to a broad range of devices. Also, this kind of interaction is very natural and easy to understand for the user.

To examine the performance of distance we have conducted two studies. The first one determined what is the optimal distance from the device for interaction and how stable (exact) is the user's input. The second one compared different ways in which the system can interpret the distance as input for selection. Here the main focus was on performance of discrete versus continuous tracking. Based on results of those studies and our literature research we propose guidelines how to optimally use distance as an input modality and how some example interfaces using it could look like.

Chapter 2

Related Work

Proximity Dependent Interfaces is a broad category encompassing different HCI topics. From Ubicomp to Around Body Interactions, through many interface types. This makes talking about them challenging because there is a multitude of devices in different shapes and sizes to analyse. That is why we decided to divide them into groups based on two important characteristics — size and interface type.

Size is an obvious categorization when discussing systems that are dependent on the physical world. The physical size of the device influences, among other things, how the device is perceived by the user, what interactions it affords and how many and what kind of tracking sensors can fit inside it.

While the *Size* category focuses on the "input" part of the interaction, the "output" part is equally important. In the case of *Proximity Dependent Interfaces*, it is usually the *Interface* that is presented to the user, that reacts in some ways to the proxemic characteristics of the user(s).

In this chapter, we analyse the *Proximity Dependent Interfaces* landscape using the above categories, at the end we explain where our work fits in.

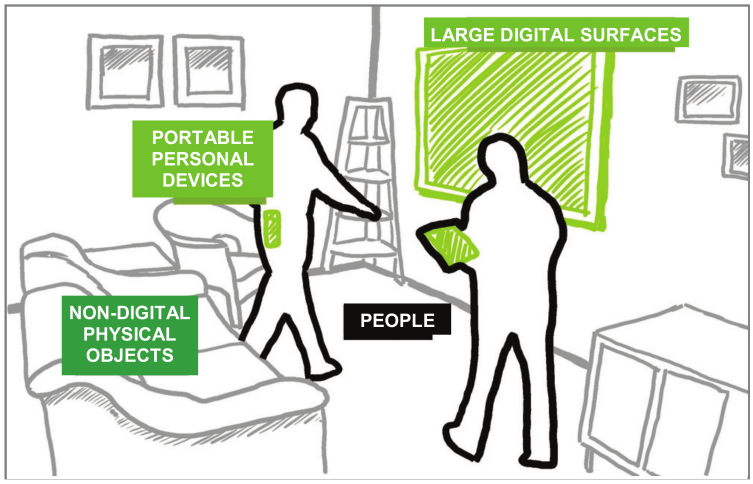

Figure 2.1: Example of a room-sized proxemic interface. Adapted from Marquardt et al. [2011]

2.1 Size

As mentioned above, the size is important because of how it influences other parts of the proxemic systems. Bigger systems usually have more and more precise sensors for tracking other entities (not only humans but also other devices as shown for example by Marquardt et al. [2012]). This results in a more natural multimodal interaction (more than one of the five proxemic factors at a time). Another difference that size introduces is the anchor of the interaction. For a room-sized system, it is usually the big central screen, while for a smartphone it is usually the user. Another variable is how many users can interact with the system at a time. The division proposed below is not strict and some of our examples could also fit in the neighbouring categories, but it will help us see how proxemic systems differ, despite being based on the same core ideas.

2.1.1 Room sized

As the name suggests, systems from this category are large. Depending on the intended use they are either a room (for

example the Ubicomp like system built by Marquardt et al. [2011]) or a public display (for example the *Hello.Wall* from Prante et al. [2003]). The anchor point of the interaction is always the big central display, capable of interacting with multiple users at once.

What all those systems have in common is the use of proxemic zones (based on Hall et al. [1966]) to control how they behave. The main idea is that different information and interactions are available at different distances from the anchor.

Some systems, like *Hello.Wall* from Prante et al. [2003] and *Interactive Public Ambient Displays* from Vogel and Balakrishnan [2004], implement a very close variation of Hall's model. Because they are in public spaces, where not only the user using the system can see the display, they use the distance to decide how personal should the shown information be.

On the other hand *The Proximity Toolkit* from Marquardt et al. [2011] is in a private space. Thus, it can use the zones to decide what kind of user interface should be shown.

Another very use case-specific interpretation of zones was proposed by Ju et al. [2008] for their interactive collaborative whiteboard. They noticed that often, during meetings, when all users move away from the board, they want to look at the bigger picture. To help with it, the board blends in notes from previous meetings. Then, when a user approaches the board, they usually want to write something new. So the board makes place for it by clustering older notes and moving them to the sides.

Of course, as we mentioned before those systems are multimodal, so although the distance is very important they also support other inputs. For example *The Proximity Toolkit* and the whiteboard support touch input, and *Interactive Public Ambient Displays* support hand gestures. Also, because those systems are often focused on collaboration, they can identify the users.

Summarizing the *room-sized* systems are multimodal systems, often used for collaboration, where the interaction is anchored around the main display.

2.1.2 Desk sized

Systems from this category are roughly the size of a desk in the same way devices from the previous category were the size of a room. They do not occupy all of this space — they use it for interaction. The most prominent differences to the previous category are that the user does not move in the space surrounding the device, rather they sit in front of it and there is usually only one user at a time.

The proxemic tracking is interpreted differently at this scale. Instead of being the main driver of the interaction, it is used to enhance standard interactions with those devices. They are not woven into the fabric of the operating system, which means they are often limited to one application that implemented them.

There are not many examples from this category, but we found two that show the focus of this category. First is *Lean and Zoom* from Harrison and Dey [2008], it uses the built-in webcam to track how far the user is from the laptop and reacts to it (described below Zoom). Important here is that the device does not track the whole user anymore, but only a part of their body (in this case the face). Other body parts are also a viable option depending on the goal of the interaction. For example *TouchCuts and TouchZoom* from Yang et al. [2011] track the distance between the user's hand and the screen to extend the interaction above the screen. Of course, it does not mean that those devices cannot track the whole body.

The *desk-sized systems* are still the anchor of the interaction, but because the interaction space is smaller the tracking is much simpler than in the previous category. And although the interfaces used right now are rather simple, there is a potential for growth if proximity tracking was added at the operating system level.

Figure 2.2: Example of Handheld proxemic interface. Adapted from Müller et al. [2015]

2.1.3 Handheld

For a long time, smaller devices did not have the power or sensors needed for meaningful proxemic interaction. They were often used as additions to *room-sized* systems as mobile terminals but were limited on their own. But now most smartphones and tablets are capable of being the centre of such interaction on their own. Also, the introduction of programmer-friendly frameworks like ARKit from Apple Inc. [2021a], made researching easier.

As Chen et al. [2014] showed, just a front-facing camera is enough to add distance and position tracking to a handheld device. Characteristic for this category is that now the user is the anchor of the interaction and because the device is smaller, it can be easily moved around them. It enables proxemic interactions that are not possible in the previous categories. For example, as Chen et al. [2012] showed in *Extending a Mobile Devices Interaction Space through Body Centric Interaction*, different functions of the device can be activated by moving it near a specific part of the body.

Similarly to *desk-sized* systems these systems also treat proxemic metrics as simple inputs, but currently, they have opposite limitations. The proxemics are used mostly for multitasking (switching applications on operating system level) and are not used inside single apps.

2.2 Interface

Although the devices differ in size, number and type of tracked inputs, most of the proxemic systems have similar interfaces. The two most popular are *Zoom* and *View Layers*, but there exist also some other use case-specific ones. As with the systems, there are more examples of interfaces from the bigger size categories, but that gives us an opportunity to analyse which of those could be useful on a smaller device.

2.2.1 Zoom

This category is divided into two kinds of zooming: view and semantic. View zooming is the kind of zooming we see when we pinch a photo on a smartphone. Part of the screen becomes bigger, and we can see the details better. It is very easy to implement, and its biggest advantage is that the users are already familiar with it. Harrison and Dey [2008] use exactly this kind of zoom to help users to lean less and have a keep posture in front of a laptop. Another example of this type of interaction can be seen in *TouchCuts and TouchZoom* built by Yang et al. [2011]. These parts of the screen got bigger as the user approached them with a finger to make selection easier.

More interesting is the semantic zoom. Now instead of seeing more detail of a single view, we see more detail of the underlying data. The simplest version of semantic zooming is described by Dostal et al. [2014] in *SpiderEyes*. They have built an interactive exhibit (wall display) that shows a map of the world with some location-specific overlaying information on it. The closer the user gets to a given part of the world, the more granular data about this part of the world is shown. It uses the fact that we naturally expect that getting closer to something reveals more details about it.

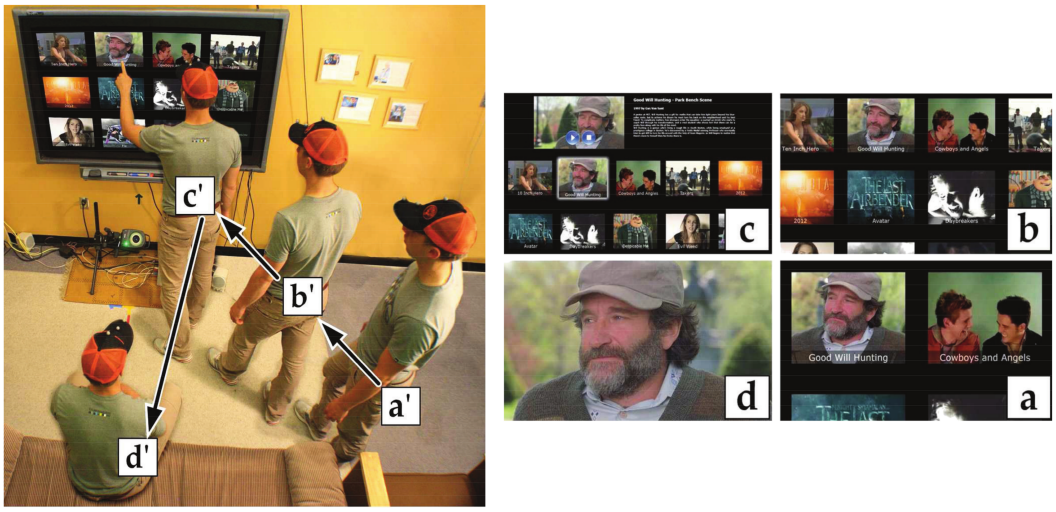

Figure 2.3: The Proximity Toolkit media player. Adapted from Ballendat et al. [2010]

The idea of zooming information can be taken a step further, instead of adding granularity to some information we can do it with the UI. We can reveal more controls or more specific actions as the user "gets closer". The best example of this is *The Proximity Toolkit* from Ballendat et al. [2010] which takes advantage of all proxemic factors described in the Introduction.

They built a media player that changes the ratio of the video player and media library on the screen depending on users distance to the screen (see Figure 2.3). When the user is far from the screen they can see few big thumbnails, that get smaller as the user gets closer. When the user is directly in front of the screen text description of a selected film appears, which would be unreadable at any other distance. The idea is that at any distance from the screen all of its contents should be readable for the user and the screen should show as much information as possible. This kind of interaction could change how we look at the notorious problem of too little screen estate on mobile devices.

2.2.2 View Layers

This kind of interface is much more popular on smaller devices, where we usually interact with only one application at a time. The idea is very simple, we have multiple views (applications or windows) open in parallel, but only one of them is visible at a time.

Again *SpiderEyes* from Dostal et al. [2014] provides us with the simplest example. The space in front of the display is divided into discrete zones, each one with a view that is visible from it. In this case, those are different types of visualization. It works similarly in the prototype from Müller et al. [2015], where the user sees different views of a map application projected on their hand depending on how close they move their hand to their body. A more advanced version of this interface is presented by Chen et al. [2014] (see Figure 2.2). Not only can the user move the device closer or further, but also to the sides and up and down, to browse even more applications more quickly.

2.2.3 Other

Here we want to mention certain interface ideas that are not as generic as the ones mentioned above, but are still worth mentioning, as they show the versatility of proxemic interfaces.

We start with *Hello.Wall* that we already mentioned before. It is an example of a truly ambient display, it does not have a usual screen, rather it is a wall of LEDs. Because it is located in a public space, only people who know how to interpret the LEDs can get the information, which in connection with the public to private zones is a great example of caring for privacy in *Proximity Dependent Interfaces*.

The second unusual system is a special display that can show different views to observers at different distances created by Dostal et al. [2013]. The benefit of such technology is that there is no need to track the users. This can be especially useful if we expect a lot of users simultaneously.

Even the authors propose it could be used at the airports or train platforms because the information the user might need directly at the gate and few meters away from it (e.g., looking for it) are different.

2.3 Relation to My Research

As we have seen, the proxemic interactions on smaller devices are not as developed as on larger devices. They mostly just use proxemics to switch between applications. We think it would be beneficial to bring interfaces using semantic zooming to mobile devices. Most beneficial would be an introduction of universal proxemic gestures, that are consistent between applications (like touch gestures currently). In this work, we want to research this topic on an abstract level. To begin with, we research what kind of switching between the zoom levels would be the best and most reliable. For this, we want to compare discrete and continuous distance tracking on handheld devices. Then based on the results propose guidelines as well as some example interfaces.

Chapter 3

Research Prototype

To conduct our research we needed a device from the personal devices' category with the ability to track the distance to the user. We decided to use an iPad 11" (2018) because it is roughly in the middle between the smallest and biggest devices from this category. It also has Apple's TrueDepth camera, which significantly simplified the implementation. This also meant, we had to choose user's face as the anchor for the distance measurement, which was in line with our interest in active interaction when the user is focused on the device.

3.1 Base Implementation

In literature, there are many implementations of distance tracking that use users face as a reference point. For example Clark et al. [2011] in *Seamless Interaction in Space* used pupil size (it was consistent between subjects) and Chen et al. [2014] in *Around-Body Interaction: Sensing & Interaction Techniques for Proprioception-Enhanced Input with Mobile Devices* used head size to calculate the distance using camera image from the front-facing camera.

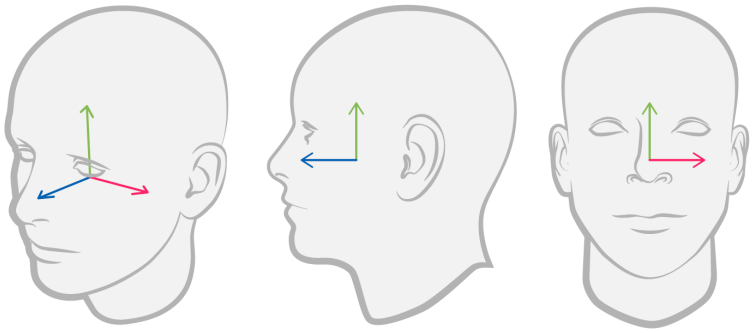

Figure 3.1: Placement of `ARFaceAnchor` in relation to users head. Adapted from Apple Inc. [2021b].

Luckily for us since iOS 11 all devices with TrueDepth camera support face tracking [1] out of the box using Apple's ARKit[2]. Through ARKit we can create an `ARSession`, which creates a 3D coordinate system with the device at its centre. When configured for face tracking this session will detect faces within 3 meters of the device's front camera. Because by default it tries to detect all faces, we had to configure it to only look for one face.

When a face is detected a 3D anchor is added to the 3D scene and its position is updated in sync with the face movement in front of the device. According to Apple's documentation, the origin of this anchor is "centred behind the face" as in the Figure 3.1. As this description is not very exact, we have decided to just trust that the framework will be consistent between different users. Because we were only interested in the distance between the face and the device, we extract it from the anchor transformation. That gives us the distance between two planes — one at the devices position and the other going through the anchor behind users face. That means that moving left, right, up or down does not influence the distance measurement.

In our experiments, we were interested in both continuous and discrete tracking of distance. So for easier reuse, we create a thin wrapper around the ARKit functionality

[1] Apple Inc. [2021b]
[2] Apple Inc. [2021a]

described above. It can be given a list of distance intervals (denoted by `ClosedRange` in Swift). Then it provides three public observable variables that tell us if the face is detected, distance to the face in meters, and in which of the provided intervals is the face.

Chapter 4

Preliminary Study

4.1 Aim & Context

Before examining more complex interactions, we wanted to determine two important constraints for distance-based interaction, because both the sensor and the user have their limitations. First was the space available for the interaction. For the sensor, based on work from Voelker et al. [2020] we expected reliable tracking between $10cm$ and $88cm$, but in our case, in some conditions, this would be additionally constrained by users' arms length.

Second constraint was the noisiness of the sensor data in relation to the distance. The sensor has some inherent noise and user is never completely still. Because of this, the accuracy of the tracking differs at different distances. This limits how accurate we can measure users position.

4.2 Apparatus

For this experiment, we created a simple app that displayed a slider (Figure 4.1). It used the continuous tracking technology described in the previous chapter. The distance between $20cm$ and $80cm$ was mapped to values from 0 to 100.

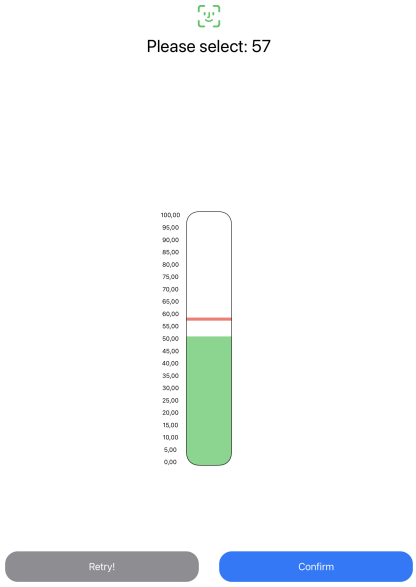

Figure 4.1: Interface of the app used for the Preliminary Study

In the STATIC condition the was placed in a stand on a table in front of the participant so that the TrueDepth camera was level with the participant's face. The participant was seated on a chair without a back, to not constraint head (and torso) movement. In condition HANDHELD the participant was seated the same way but was asked to hold the tablet in their hands above the table, without resting them on it. The participant was also instructed to use head movement when possible and only move the torso or hands if necessary. The starting distance for the experiment was $40cm$, which was guaranteed by the start screen in the app.

4.3 Participants

Because it was just a pilot study (and because of the pandemic situation) we only had six participants (20-28 years, $M = 23.5, SD = 2.6$). All have a technical background but have never used proximity as an input method before. To avoid the learning effect half of the users started in condition STATIC and the other half in condition HANDHELD.

4.4 Design & Task

The experimenter prepared the app on the iPad and depending on the condition either placed it on the stand or gave it to the participant. The participant moved to the start position ($40cm$ from the screen) with help of the start screen and started the experiment when they were ready.

During the experiment the app showed to the user what value they should select. After each selection, the participant had to hold still for two seconds. In this time we measured the minimal, maximal and average values of their selection. After the time was up new value was shown and the process repeated until all values were selected.

To assure consistency between the participants, target values were always the same 12 values $(0, 10, 16, 23, 28, 35, 40, 47, 57, 63, 74, 83, 90, 98)$ in random order. Each value had to be selected four times, which gave us 48 selections per participant per condition. Which took about five minutes.

4.4.1 Independent Variable

The only thing that we varied in this experiment was the position of the iPad. It was either standing on a table (STATIC) or was held by the participant (HANDHELD).

4.4.2 Dependent Variables

We are measuring the distance of participant's head to the device in the space between $20cm$ and $80cm$ from the device. We measured the values of the slider but later in the results section following variables are reported in cm.

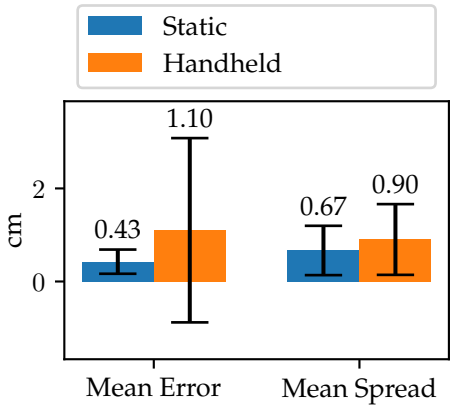

Figure 4.2: Mean Error: mean difference between the desired value and the average value selected by the user in cm. **Mean Spread**: mean of the difference between the smallest and biggest value around the target value during the 2 sec. selection. Both values showed with standard deviation.

ERROR This was defined as the difference between the target value and the average value registered over the two seconds wait period. Because the direction of the error had no meaning to us, we measured the absolute value.

SPREAD This was the difference between the maximal and minimal value registered over the two seconds wait period. It is an estimated value of how big a discrete layer of the available space should be.

4.5 Results

For all target values the STATIC condition caused a smaller mean error ($0.43cm$, $SD = 0.26cm$) and a smaller mean interval ($0.67cm$, $SD = 0.53cm$). Condition HANDHELD had respectively $1.1cm$ ($SD = 1.98cm$) and $0.9cm$ ($SD = 0.76cm$). Visible in Figure 4.2.

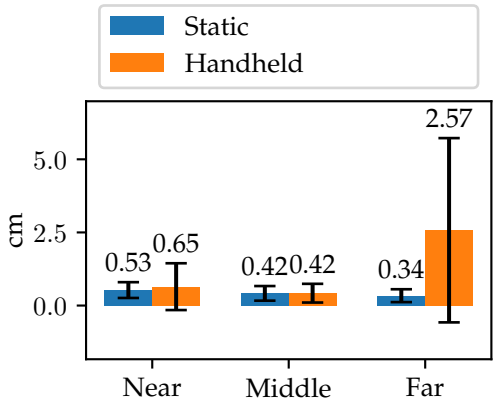

Figure 4.3: Average difference between desired value and selected one in cm with standard deviation for each distance group.

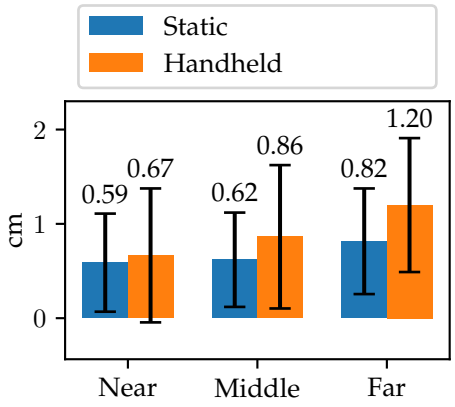

Figure 4.4: Mean of difference between biggest and smallest value around the target value during hold time for each distance group.

When analysing the collected data, we noticed an interaction effect between the condition and the distance. The participants had most problems with selecting the values nearest and furthest from the screen. So for further evaluation, we divided the target values into three categories: NEAR $(0, 10, 16, 23)$, MIDDLE $(28, 35, 40, 47, 57, 63)$, FAR $(74, 83, 90, 98)$. The differences between them are visible in figures 4.3 and 4.4. Most interesting is that in the MIDDLE group the difference between mean errors was negligible $(0.006cm)$, although in all other cases HANDHELD had worse performance.

4.6 Evaluation

As we expected, overall in the STATIC condition the accuracy was higher than in HANDHELD. Because the only source of error was the unintentional head movement when the user tried to stay still.

But when we exclude the extreme values (NEAR, FAR) the differences become relatively small $(0.006cm)$. In the case of the NEAR section, we suspect the bigger error is caused

by the way the ARKit tracks the face. When the user gets too close to the device, their face is bigger than the field of view of the camera and the framework has a difficult time finding and tracking the face. That is why this mean error is bigger both for STATIC and HANDHELD compared to MIDDLE. For the FAR section the mean error gets bigger only in the HANDHELD condition, so it is probably connected with it. We suspect here the max distance is constrained by users arms' lengths.

Some users tried to circumvent this problem by tilting the device with arms stretched. Asked about it, most participants stated that it was not very comfortable, but they wanted to get the value anyway. When asked if they could see the screen they stated that it was hard, some participants also added that in the closest position the screen was not fully visible either.

Based on this study we recommend using the middle of the available space for the interaction, as provides both best tracking and is comfortable for the user. In our case it is the space $37 - 57cm$ from the available $20 - 80cm$, but it may vary based on the tracking setup, screen size, etc. Unless the discomfort is used deliberately, to signify to the user that the action is somehow special [4].

Our second recommendation concerns the measured interval sizes. We have seen that the average interval in which the user stays while trying to hold still is around one centimetre and the biggest one was $3.7cm$. So for least error we recommend using at least $4cm$ targets, but everything over $2cm$ should be usable. That means in our interval from previous paragraph, we could fit five to ten discrete layers.

Chapter 5

Comparison of Techniques to Specify Values using Proximity Input

After the previous study, we know *where* to place the interaction to minimize tracking errors. Now we want to find out *how* different types tracking perform in this space. Because of its physical nature distance is measured linearly, but that does not mean it can only be used for such inputs. It is also very popular to track the distance using discrete zones.

But those methods have one big disadvantage — they always reset the current selection and are limited to their finite domain. That is why we also added a relative control in form of a joystick to this comparison. As the name suggests, it can be used to traverse infinite linear intervals at different speeds starting from the current value.

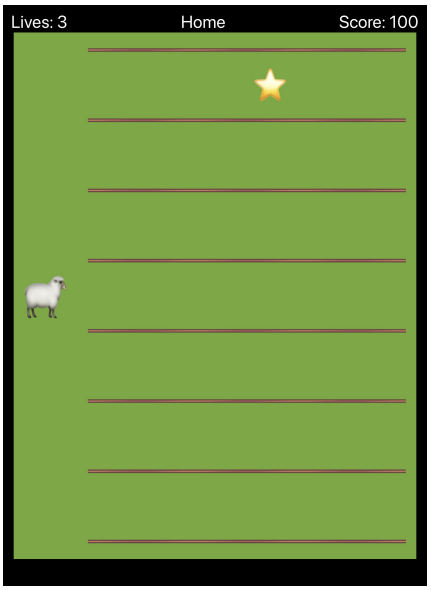

Figure 5.1: Interface of the app used for the Main Study

5.1 Aim & Context

In this study, our goal was to evaluate those different mappings. Because how they perform, implies for what interactions distance based control can be used. Additionally, we wanted to see, if the user can reliably assess the distance to the device.

This time we did not differentiate between static and handheld, because we placed the controls in the MIDDLE section of the available space. Based on our Preliminary Study we knew that the results could be generalized for both positions without much error.

5.2 Apparatus

To make this study more enjoyable for the participants and to get them more invested we decided to create a simple catching game (Figure 5.1) using the base described in the Prototype Chapter. During the game the rewards appear

on the right side of the screen and move to the left, where the participant should catch them. As the game progresses, the rewards get faster (REWARDSPEED). The way the sheep character moves depends on the MAPPING condition.

The iPad was placed on a stand so that the TrueDepth camera was level with the participant's face. The starting position was again $40cm$ from the screen. Because now the interaction space was only $12cm$ big ($0.34cm - 0.46cm$), we used a chair with a back (ending below the neck) to minimize torso movement.

5.3 Participants

For this study we recruited 15 participants aged between 20 and 28 ($M = 22, SD = 2.26$, 1 female), four of which already participated in the first study, the rest did not have any previous experience with proximity-based interfaces before. We used within groups design to account for different skill levels of the participants (3 do not play any games at all, 5 play more than five times per week). To avoid the learning effect we used Latin Square to randomize in which order participants played the levels.

5.4 Design & Task

During each trail the experimenter first selected the correct mapping and then placed the iPad on a stand in front of the participant, who moved to the start position ($40cm$ from the screen) and pressed the *Start* button.

After that, they entered the level and had time to test how the tracking method works. It was important that the participant understood how the sheep would move depending on their movement before the trial began. The participant was free to start when they felt ready by pressing a *Play* button. No participant needed more than 2 minutes for this.

Now the game started, and the rewards started to appear. The user moved their head closer or further to the screen to move the sheep to the row with the star to collect it. After 91 stars were spawned and their speed went down to $0.5s$ the life counter became active. Each not caught star did cost one life, so after three misses the game ended.

5.4.1 Independent Variables

REWARDSPEED

As the game progresses the rewards move faster from right to left. At the beginning they need $3sec$ to move from right to left. Then every 14 or 21 rewards they get $0.5sec$ faster. Although the game continues when the REWARDSPEED is $\leq 0.5sec$, we do not consider those values in our study.

MAPPING

There are three main categories of mappings: DISCRETE, CONTINUOUS, JOYSTICK.

DISCRETE In this condition the control space ($12cm$) is divided into 3, 5 or 7 layers (accordingly $4cm, 2.4cm, 1.7cm$ thick). We have chosen those sizes based on our Preliminary Study. There we have seen that the biggest interval was $3.63cm$, so in $4cm$ thick variation the participant should be able to not miss any rewards.
Later we refer to those conditions as D3, D5, D7.

CONTINUOUS In this condition we have two variations: LINEAR and NONLINEAR.

LINEAR control is a linear translation of heads position to sheep's position. Where head at $34cm$ is equivalent to sheep at the top of the screen and at $46cm$ to sheep at the bottom of the screen.

In NONLINEAR the translation uses a tan function ($y = 0.5 * tan(1.8 * x)$) to move the sheep. This results in near-linear tracking in three middle rows and faster character movement in four outer rows. It means the control in the middle is more precise and the outer values are easier to reach. Resulting in less intense head movement.
Later we refer to those conditions as CL, CC.

JOYSTICK Both previous mappings are absolute, i.e., the output is limited to some interval. But in real-life situations there is also a need for infinite output (e.g., when scrolling an infinite timeline). For this purpose we created a joystick-like control where the participant's head functions as the joystick's knob. That means if the participant moves their head forward the character starts moving up at a constant rate and the more forward the participant moves the faster the character moves. And analogous for moving the head backwards.

Here we also have two variations: LINEAR and NONLINEAR, which are similar to ones from CONTINUOUS condition. LINEAR functions as described above and NONLINEAR adds space in the middle which allows stopping the movement of the character.
Later we refer to those conditions as JL, JC.

5.4.2 Dependent Variables

COLLECTEDRATIO We tracked how many rewards the participant collected per control type and per reward speed. Because the number of rewards per speed was different — fewer rewards at a lower speed, more at higher — we divided the collected amount by the spawned amount to get the COLLECTEDRATIO.

TIMETOREACH This metric tells us how much time did it take for the participant to move the character to the row with the reward after a new reward appeared.

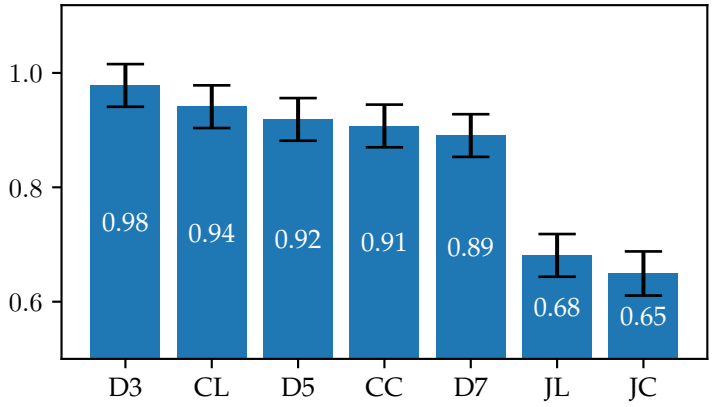

Figure 5.2: CollectedRatio per condition with confidence intervals. The bigger the number the more rewards did the user collect.

OvershootRatio This metric tells us how stable could the participant hold the position inside a row with the reward. We started counting over- and undershoots after the character first reached the row with the reward. Overshoot counted when the participant moved out of the row with reward to such extent that they would not be able to collect the reward if it was at sheep's horizontal position.

Similarly to CollectedRatio this count was normalized by dividing it by the number of awards per speed. That means the overshoot ratio can be bigger than one, i.e., multiple overshoots for one reward are possible.

5.5 Results

5.5.1 Prox mity

CollectedRatio We analysed CollectedRatio against Condition using one way ANOVA. There is a significant difference between the seven conditions ($F(6, 513) = 46.40$). This means that the mapping used influences the number of collected rewards.

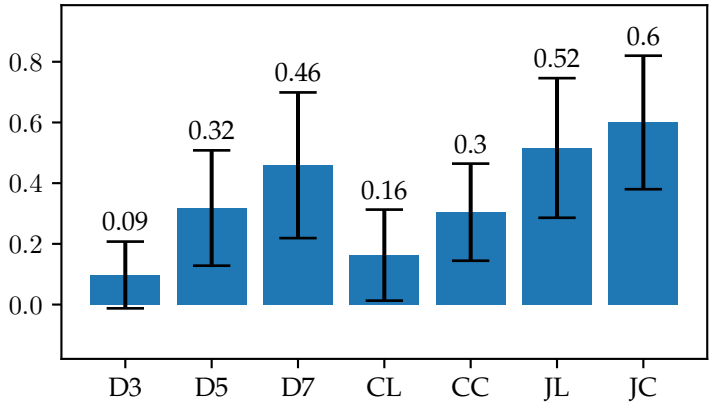

Figure 5.3: OVERSHOOTRATIO ratio per condition, with standard deviation. The smaller the number, the easier could the user stay in row with the reward.

To know exactly how the conditions differed we then conducted Student T-Tests for each pair of conditions. We have got three groups that differ significantly from each other.

First interesting observation is that the most basic variants of DISCRETE ($\overline{\text{D3}} = 0.98$) and CONTINUOUS ($\overline{\text{CL}} = 0.94$) conditions did not differ significantly. But D3 differs significantly from other DISCRETE conditions, which would suggest that if we have more levels the CONTINUOUS tracking is preferable to DISCRETE.
Because they did not differ significantly ($\overline{\text{CL}} = 0.94$, $\overline{\text{CC}} = 0.91$, $\overline{\text{D5}} = 0.92$, $\overline{\text{D7}} = 0.0.89$). Both JOYSTICK variants scored significantly worse than all other conditions ($\overline{\text{JL}} = 0.68, \overline{\text{JC}} = 0.65$), but they did not differ significantly from each other.

OVERSHOOTRATIO We analysed OVERSHOOT against CONDITION using one way ANOVA. Because the collected data was not normally distributed, we transformed it using the log function to do the calculations. We found a significant difference between the seven conditions ($F(6, 513) = 76.67$). This means the mapping method influences the number of overshoots and undershoots.

To know exactly how the conditions differed we then conducted Student T-Tests for each pair of conditions. We have got three groups that differ significantly from each other.

All DISCRETE variants differed significantly from each other. As we expected based on the Preliminary study, the bigger the interval was, the fewer overshoots it had ($\overline{D3} = 0.09, \overline{D5} = 0.32, \overline{D7} = 0.46$). D7 was significantly worse than both CONTINUOUS variants ($\overline{CL} = 0.16, \overline{CC} = 0.30$). And again the JOYSTICK performed worse ($\overline{JL} = 0.52, \overline{JC} = 0.60$). However, D7 was not significantly better than JL.

TIMETOREACH First we analysed TIMETOREACH against REWARDSPEED (the time the reward had to travel from right to left). Across all conditions TIMETOREACH was always between one third and a half of the REWARDSPEED. Which shows that when the participant had less time they reacted faster, but as we can see in Figure 5.4, simultaneously the COLLECTEDRATIO went down.

Next, we evaluated TIMETOREACH against conditions for each REWARDSPEED using ANOVA. For all REWARDSPEEDs there were significant differences between the conditions, but for most the effect sizes were negligible. Only at $1s$ there was an important difference. Both JOYSTICK variants were two times worse than the average of all other controls ($\overline{J} = 0.56$ vs $\overline{rest} = 0.27$).

5.5.2 NASA TLX

Additionally, to sensor data collected through the app, we also wanted to quantify the workload connected with each of the seven conditions. For this, we used a multidimensional rating procedure: NASA TLX. It consists of six factors: *Mental Demand*, *Physical Demand*, *Temporal Demand*, *Performance*, *Effort*, *Frustration*. Each of them is measured on a scale from 0 to 100. Below we report on two factors that showed some significant differences between conditions.

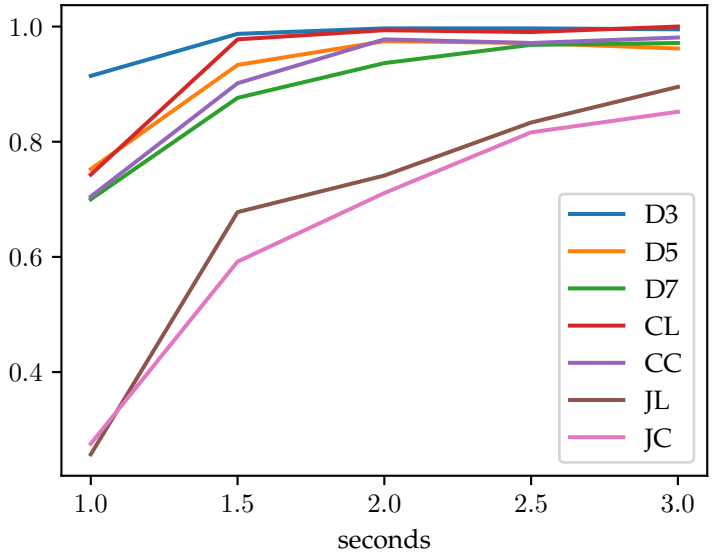

Figure 5.4: CollectedRatio per RewardSpeed
Shows how change of time to react influenced the amount of collected rewards.

Effort Here the lower score means that less effort was required for the interaction. We first analysed with one-way ANOVA. There is a significant difference between our seven conditions ($F(6, 96) = 3.10$). To see where the significance is, used Student T-Test to compare all pairs of conditions.

Like in CollectedRatio D3 ($\overline{\text{D3}} = 32.86$) and CL($\overline{\text{CL}} = 49.29$) did not significantly differ from each other. But together they differed significantly from all other conditions together.

Again Joystick was rated as requiring most effort with $\overline{\text{JC}} = 68.67, \overline{\text{JL}} = 67.34$. However, it did not differ significantly from D5 ($\overline{\text{D5}} = 52.34$), D7 ($\overline{\text{D7}} = 55.00$) and CC ($\overline{\text{CC}} = 54.67$).

Frustration Here the higher score means the more the participant was frustrated during the interaction. Analysed

with one-way ANOVA. There is a significant difference between our seven conditions ($F(6, 96) = 3.64$). To see where the significance is, we used Student T-Test to compare all pairs of conditions.

In each of the three input CONDITION the variants did not differ significantly. Again CL ($\overline{\text{CL}} = 27.14$) and D3 ($\overline{\text{D3}} = 34.64$) caused the least frustration and JOYSTICK caused the most $\overline{\text{JC}} = 70.67, \overline{\text{JL}} = 63.4$. But they did not significantly differ from D7 ($\overline{\text{D7}} = 48.34$).

5.6 Evaluation

Looking at all results from this experiment we can see an overwhelming trend. The simpler the control, the better it performed. This overlaps with the statement of Harrison and Dey [2008], which used controls similar to our CL, that the learning curve is minimal. But it did not hold for the less intuitive conditions. We suspect that novelty is partially responsible for the weak performance of both JOYSTICK variants. Many participants stated that the joystick felt unintuitive and confusing, although some of them were familiar with joysticks in game controllers.

Another thing many participants complained about was the lack of feedback in DISCRETE and JOYSTICK conditions. They were not sure how far do they have to move the change layer (row). This could explain why overall the CONTINUOUS tracking method scored the best, even beating D5 and D7.

Because JOYSTICK performed so poorly in this study, we recommend using the other two mappings. This means if we wanted to select a value we would always need to start from the start position, as discrete and continuous tracking methods lack the ability to start from current value. So, we do not recommend using distance for this kind of implicit input.

As we have seen in the Preliminary Study, the available space can be divided into 5-10 discrete layers. So the dis-

tance tracking could be used to select something from a context menu with up to 10 elements, because in this scenario we do not have a previous selection. This idea is explored in detail Enhancing dropdown experience.

That does not mean we think the joystick idea is completely wrong. We believe it would be beneficial to evaluate its different implementation in future work, for example one with some additional activator.

Independent of the use case it is very important to provide appropriate feedback, informing the user where they currently are in the control space. Also, ideally, there should not be a time limit for selection and if there has to be one, it should take into account the number of selections.

Chapter 6

Discussion

6.1 Example use cases

Although we conducted our research using a tablet (a handheld device), we believe that our findings also apply to interactions with laptops or even desktop computers. That is why we provide examples both for a laptop and a smartphone.

6.1.1 Enhancing multitasking experience

All the most popular desktop operating systems already offer multitasking in form of multiple windows and desktops. But it requires the use of keyboard shortcuts or gestures to access those multitasking views.

We want to focus on two of those multitasking views available in macOS: App Exposé and Mission Control. The first shows us all windows of one application, while the other shows all open applications and desktops. In other words, one helps us get closer to one application and the other one gives us an overview of the whole system. So staying true to the intuitiveness of proxemic design, leaning in could activate App Exposé and lean back could activate the Mission Control.

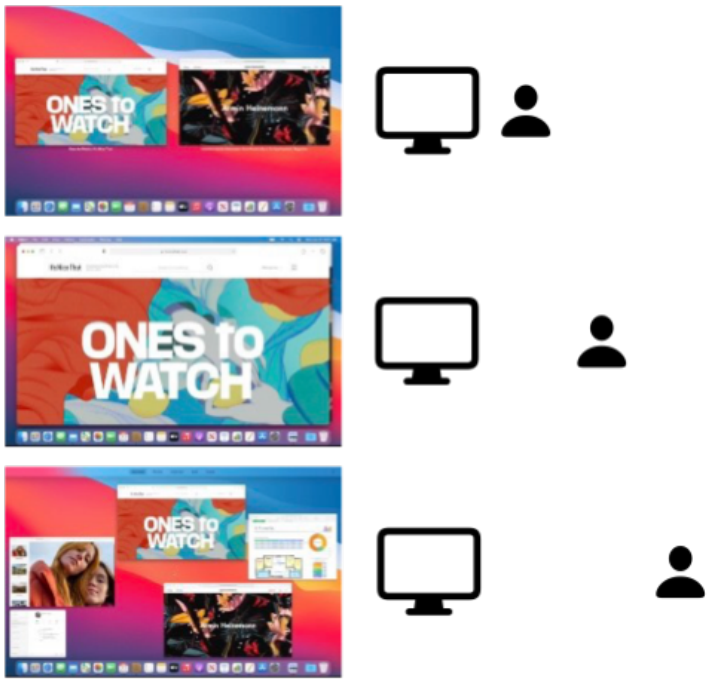

Figure 6.1: Example how screen contents could change when the user gets close or further. The default is the middle where the user sees the standard desktop.

As we have shown in the second study, at this scale three discrete states should function without any problems. The three layers should also be sufficiently large to allow for some natural head movement without triggering any action. But it is very important to signal to the user when they are approaching a change in state. Indicator constantly showing in which of the three layers the user is, could be a solution but is not ideal. It uses up screen real estate (this can be a concern on smaller laptops) and does not provide useful information most of the time, assuming the user spends most time in the middle layer (standard desktop). That is why we propose to show a semitransparent colour frame around the screen when the user is less than one centimetre from the layer threshold.

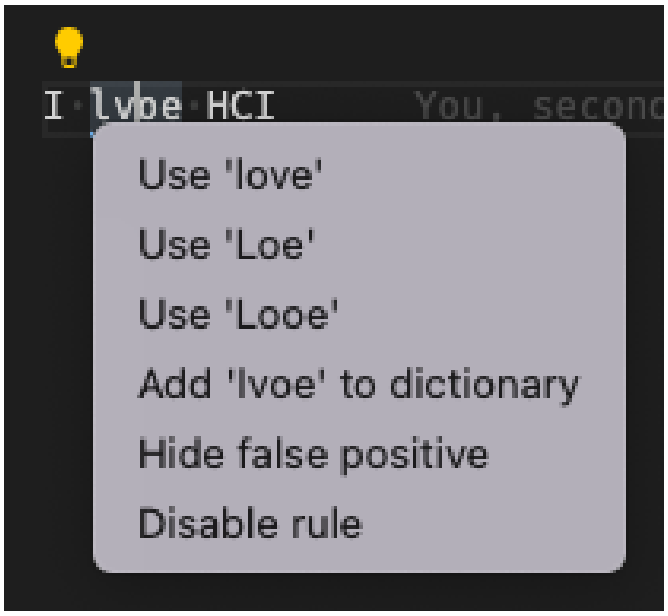

Figure 6.2: Example dropdown from VS Code IDE

6.1.2 Enhancing dropdown experience

Another conclusion from our studies is that distance can be reliably used to select an element from a short list when there is no time pressure. That is a use case very common in many IDEs for inline code actions (for example Figure 6.2). Currently, depending on IDE, to open the dropdown the user can either click the lightbulb or use a keyboard shortcut. Then they have to use either mouse or arrow keys to select the desired action. Both actions require moving hands away from the keyboard which makes the whole interaction longer (based on GOMS model from Card et al. [1983]). We propose to instead use the linear continuous tracking (see Comparison of Techniques to Specify Values using Proximity Input) to select the option, while the user is holding the keyboard combination. This presents a small problem of closing the menu without selecting anything. For this we can use the orientation of the user's head — it is natural to turn away from things that do not interest us.

We see that the above examples function on different distance scales. One uses leaning of the torso and the other slight movement of the head. Because of this, they can be

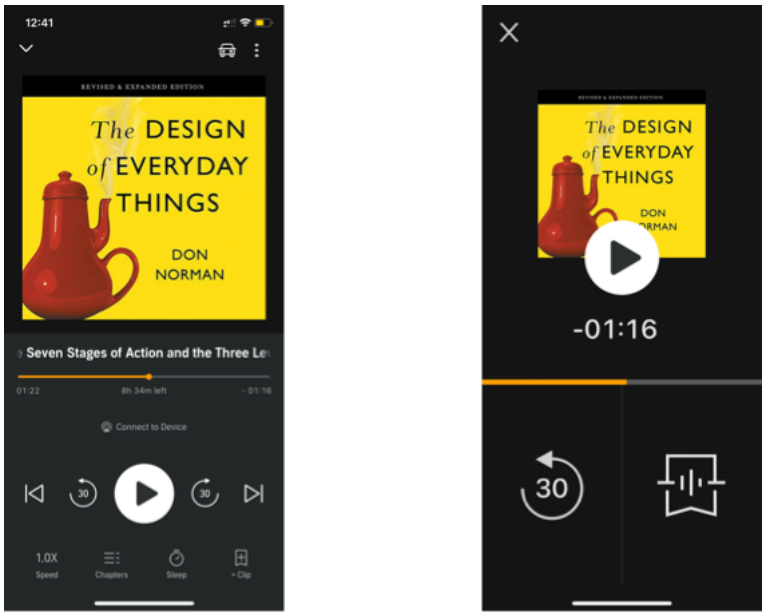

Figure 6.3: Example of currently existing mobile UI that can be changed depending on how far the phone will be from the user. Screenshots from Audible app for iOS.

used at the same time without interfering with each other. This shows that distance-based interactions can coexist in one system as long as they use different scales or activators.

6.1.3 Smartphone Adaptive view

As we have seen in Related Work, most mobile devices which use proxemic tend to use it for multitasking switching between apps. But the idea of changing UI in one app does not seem that popular, unless we look at already available apps that do not use proxemics, but just assume they will be used at a distance. Apps intended for use while driving often include a simplified screen with only essential controls. For example, the audiobook app Audible (see Figure 6.3) reduces the number of buttons (13 to 4) and makes the buttons fill the whole available space. The goal is to minimize both distractions and the time the user has to look for the button to make driving safer.

But this design has one big flaw to get to this simplified state the user must first find and click the right button. So going back to our car example, if the user, while waiting at a red light, got closer to their phone and switched back to the detail view, and it suddenly turned green, they will be stuck in the detail view, although they moved back from the phone. The advantage of using proximity for switching between this kind of detail and the essential view is clear. It might even allow for some state between the two extremes.

What would be even better is an interface that can zoom semantically. It would animate appearing a disappearing of controls, to avoid confusing the user. As Brock et al. [2018] have already shown, just showing and hiding parts of the UI can be missed by the user. Ultimately when the users get used to this kind of scaling of controls, they might even start expecting this kind of three-dimensional interaction.

Chapter 7

Summary and future work

In this work, we focused on *Proximity Dependent Interfaces* on small and medium devices. After analysing the current landscape of *Proximity Dependent Interfaces*in literature, we noticed that this niche is underexplored. So we wanted to apply the interfaces seen on larger proxemic devices to the smaller ones.

7.1 Summary and contributions

In this work, we have made three main contributions. First, we established the ideal distance for distance-based interactions and found how accurate is an average user. Secondly, we evaluated different distance mapping for selection techniques. And based on this we propose which of them are best to use in what scenario. Lastly, we proposed three new types of distance interactions, that can be implemented on current devices.

7.2 Future Work

Because of time and volume limitations posed by this kind of thesis we only managed to scratch the surface of *Proximity Dependent Interfaces* on smaller devices. We think the

JOYSTICK mapping was at a disadvantage in this study. So we would like to examine it once again in a task that could take advantage of its infinite, relative control. This time instead of constant tracking, we would like to use some activator — touch or keypress — that would allow the user to move freely when not using the control. Also, ideally, we would be able to test it over a longer period of time, to eliminate the influence of novelty.

Another aspect that we did not consider in this work was using proximity on the go. Interacting with mobile devices when moving is a common problem in , so to show that proxemics is a viable input modality for mobile we want to also evaluate how it performs there.

There is also a non-technical aspect that concerns more and more users — privacy. Many participants during both studies asked the experimenter if they were being recorded. We think, surveying potential users to learn what types of tracking they find acceptable, is an important part of responsible research.

Bibliography

Apple Inc. Apple arkit, 2021a. URL https://developer.apple.com/augmented-reality/arkit/.

Apple Inc. Apple arkit facetracking, 2021b. URL https://developer.apple.com/documentation/arkit/arfaceanchor.

Till Ballendat, Nicolai Marquardt, and Saul Greenberg. Proxemic interaction: Designing for a proximity and orientation-aware environment. In *ACM International Conference on Interactive Tabletops and Surfaces*, ITS '10, page 121–130, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450303996. doi: 10.1145/1936652.1936676. URL https://doi.org/10.1145/1936652.1936676.

Steve Benford, Chris Greenhalgh, Gabriella Giannachi, Brendan Walker, Joe Marshall, and Tom Rodden. *Uncomfortable Interactions*, page 2005–2014. Association for Computing Machinery, New York, NY, USA, 2012. ISBN 9781450310154. URL https://doi.org/10.1145/2207676.2208347.

Michael Brock, Aaron Quigley, and Per Ola Kristensson. *Change Blindness in Proximity-Aware Mobile Interfaces*, page 1–7. Association for Computing Machinery, New York, NY, USA, 2018. ISBN 9781450356206. URL https://doi.org/10.1145/3173574.3173617.

Stuart K. Card, Allen Newell, and Thomas P. Moran. *The Psychology of Human-Computer Interaction*. L. Erlbaum Associates Inc., USA, 1983. ISBN 0898592437.

Xiang 'Anthony' Chen, Nicolai Marquardt, Anthony Tang, Sebastian Boring, and Saul Greenberg. Extending a mobile device's interaction space through body-centric interaction. In *Proceedings of the 14th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '12, page 151–160, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450311052. doi: 10.1145/2371574.2371599. URL https://doi.org/10.1145/2371574.2371599.

Xiang 'Anthony' Chen, Julia Schwarz, Chris Harrison, Jennifer Mankoff, and Scott Hudson. Around-body interaction: Sensing & interaction techniques for proprioception-enhanced input with mobile devices. In *Proceedings of the 16th International Conference on Human-Computer Interaction with Mobile Devices & Services*, MobileHCI '14, page 287–290, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450330046. doi: 10.1145/2628363.2628402. URL https://doi.org/10.1145/2628363.2628402.

Adrian Clark, Andreas Dünser, Mark Billinghurst, Thammathip Piumsomboon, and David Altimira. Seamless interaction in space. In *Proceedings of the 23rd Australian Computer-Human Interaction Conference*, OzCHI '11, page 88–97, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450310901. doi: 10.1145/2071536.2071549. URL https://doi.org/10.1145/2071536.2071549.

Jakub Dostal, Per Ola Kristensson, and Aaron Quigley. Multi-view proxemics: Distance and position sensitive interaction. In *Proceedings of the 2nd ACM International Symposium on Pervasive Displays*, PerDis '13, page 1–6, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450320962. doi: 10.1145/2491568.2491570. URL https://doi.org/10.1145/2491568.2491570.

Jakub Dostal, Uta Hinrichs, Per Ola Kristensson, and Aaron Quigley. Spidereyes: Designing attention- and proximity-aware collaborative interfaces for wall-sized displays. In *Proceedings of the 19th International Conference on Intelligent User Interfaces*, IUI '14, page 143–152,

New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450321846. doi: 10.1145/2557500.2557541. URL https://doi.org/10.1145/2557500.2557541.

Saul Greenberg, Nicolai Marquardt, Till Ballendat, Rob Diaz-Marino, and Miaosen Wang. Proxemic interactions: The new ubicomp? *Interactions*, 18(1): 42–50, January 2011. ISSN 1072-5520. doi: 10.1145/1897239.1897250. URL https://doi.org/10.1145/1897239.1897250.

E.T. Hall, E.T. Hall, and Copyright Paperback Collection (Library of Congress). *The Hidden Dimension*. Anchor books. Doubleday, 1966. ISBN 9780385084765. URL https://books.google.de/books?id=0mNrkzvv8y4C.

Chris Harrison and Anind K. Dey. Lean and zoom: Proximity-aware user interface and content magnification. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, page 507–510, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605580111. doi: 10.1145/1357054.1357135. URL https://doi.org/10.1145/1357054.1357135.

Wendy Ju, Brian A. Lee, and Scott R. Klemmer. Range: Exploring implicit interaction through electronic whiteboard design. In *Proceedings of the 2008 ACM Conference on Computer Supported Cooperative Work*, CSCW '08, page 17–26, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605580074. doi: 10.1145/1460563.1460569. URL https://doi.org/10.1145/1460563.1460569.

Nicolai Marquardt and Saul Greenberg. Informing the design of proxemic interactions, 2011. URL https://prism.ucalgary.ca/handle/1880/48690.

Nicolai Marquardt, Robert Diaz-Marino, Sebastian Boring, and Saul Greenberg. The proximity toolkit: Prototyping proxemic interactions in ubiquitous computing ecologies. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, page

315–326, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450307161. doi: 10.1145/2047196.2047238. URL https://doi.org/10.1145/2047196.2047238.

Nicolai Marquardt, Till Ballendat, Sebastian Boring, Saul Greenberg, and Ken Hinckley. Gradual engagement: Facilitating information exchange between digital devices as a function of proximity. In *Proceedings of the 2012 ACM International Conference on Interactive Tabletops and Surfaces*, ITS '12, page 31–40, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450312097. doi: 10.1145/2396636.2396642. URL https://doi.org/10.1145/2396636.2396642.

Florian Müller, Mohammadreza Khalilbeigi, Niloofar Dezfuli, Alireza Sahami Shirazi, Sebastian Günther, and Max Mühlhäuser. A study on proximity-based hand input for one-handed mobile interaction. In *Proceedings of the 3rd ACM Symposium on Spatial User Interaction*, SUI '15, page 53–56, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450337038. doi: 10.1145/2788940.2788955. URL https://doi.org/10.1145/2788940.2788955.

Thorsten Prante, Carsten Röcker, Norbert Streitz, Richard Stenzel, Carsten Magerkurth, Daniel Van Alphen, and Daniela Plewe. Hello. wall–beyond ambient displays. In *Adjunct Proceedings of Ubicomp*, volume 2003, pages 277–278, 2003.

Simon Voelker, Sebastian Hueber, Christian Holz, Christian Remy, and Nicolai Marquardt. *GazeConduits: Calibration-Free Cross-Device Collaboration through Gaze and Touch*, page 1–10. Association for Computing Machinery, New York, NY, USA, 2020. ISBN 9781450367080. URL https://doi.org/10.1145/3313831.3376578.

Daniel Vogel and Ravin Balakrishnan. Interactive public ambient displays: Transitioning from implicit to explicit, public to personal, interaction with multiple users. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*, UIST '04, page 137–146, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581139578. doi: 10.1145/

1029632.1029656. URL https://doi.org/10.1145/1029632.1029656.

Xing-Dong Yang, Tovi Grossman, Pourang Irani, and George Fitzmaurice. *TouchCuts and TouchZoom: Enhanced Target Selection for Touch Displays Using Finger Proximity Sensing*, page 2585–2594. Association for Computing Machinery, New York, NY, USA, 2011. ISBN 9781450302289. URL https://doi.org/10.1145/1978942.1979319.

Index

Typeset September 1, 2021