UNIVERSITY OF COPENHAGEN
FACULTY OF SCIENCE

Master's thesis, Department of Computer Science
University of Copenhagen, fall 2016
Supervisor: Sebastian Boring

Off-screen Interactions Beyond the Display Boundaries of Portable Devices

Rune Jensen

Abstract

Much of today's human-computer interaction takes place on portable displays that are based solely on using touch as input. However, since smaller displays are not able to present much information, navigational techniques must be available for the user to efficiently navigate a potentially large information space, which may only be partially in view. But if these techniques rely on touch alone, it creates a similar problem, because the display plane itself provides very limited input space and has an easily obstructed view. What results are redundant inputs for invoking highly similar actions, possibly with a high use of pseudo-input (such as inertia) for maintaining a clear line of view. Hence, if a mobile touch-based interface could extend beyond its input boundaries and into an imaginary extension of the display plane, it would likely eliminate much of this input redundancy, leading to fewer touch operations and presumably open up for more efficient navigation of the user interface.

Using three-dimensional space for input has recently become a highly active research field, one with great potential in the multitude of cataclysms that are currently unfolding in all areas of computer science. The paper here contributes to the research body with an exploration of one potential usage strategy for exploiting the space surrounding a mobile device, done so through a solution based on related work and implemented using current technology. To measure up its applicability, test subjects perform a sequence of navigational tasks that incorporates this space and the result is compared to a baseline that does not. What is seen, is that such baseline is hard to beat in terms of interaction time, but that one particular definition of the off-screen space comes very close and in addition, greatly enhances the overall user experience. Resulting questions are then explored further. The test subjects' perceptions of off-screen space are evaluated through another experiment and the results subsequently used to modify the original solution. A final experiment then shows how this brings about a modest performance improvement in interaction time, when compared to the original solution, as well as potential pathways to take in future work.

Contents

1 Introduction

The use of tools is a trademark of human history, both in simple times of mere survival and even more so in the establishment of societies and specializations of trades. This potential for people to gain from optimization through specialization is not the result of any obvious physical superiority. Quite the contrary, its stems from the ability to communicate abstract concepts over time, with the tool itself being nothing more than a momentary manifestation.

While the concept of a tool is abstract, its physical realization always takes some mechanized form intended for human interaction, one that usually that draws on our most distinguished evolutionary advantage: the extraordinary flexibility and sensitivity of our hands and fingers. The computer is well-suited in this regard, due the lack of mechanics. The most remarkable human tool to date, it is capable of emulating the same logic that is embedded in all natural laws, but logic is all information and has no matter or motion. The application of logic, through algorithms that process information, results in nothing but information itself. Not surprisingly, the physical activity needed to interact with a computer is also limited to information, provided through alphabetized languages for defining computational steps.

The most successful input methods reflect these observations, in the sense that they rely on the hands to provide a vast array of combinatorial input with as little mechanical action as possible. In addition, the flexibility of modern displays help to rapidly comprehend informational complexity, due to the fast perceptual speed of vision. Display technology has recently changed the way we interact with computers, due to quantum leaps in terms of quality, dimensionality and price reduction. As an obvious real-world example, mobile telephones went from panels of knobs and buttons to all display-based miniature computers almost overnight - *mobile devices* that incorporate a multitude of functions, each previously only available in some dedicated and more mechanical form factor. In other words, rapid progress in one technology facilitated the convergence of a multitude of others, such that a modern lifestyle now includes having a mobile device - using a pocket-size rectangular display to perform a wide variety of daily activities is now the norm. Not surprisingly, developments in user interface design principles have followed suit, with designers unanimously adopting a *Mobile First*[27] paradigm that anticipates the most probable human-computer interaction as one based on mobile devices.

Although these minimalistic devices provide mobility, there are caveats. Designing an appropriate navigational user interface on touch-based devices cannot be approached with a one-size-fits-all paradigm, but must take into account some basic factors that greatly influences the user experience. As an imaginary example, a user dragging a map by touch on a large display will typically be positioned close to the center of the screen. Hence, with half or more of the displayed map in peripheral view, the user then has a good overview of the information space. This will obviously not apply in the case of a small portable device, where conditions are opposite - the display being in full view, but the presentation of input space severely restricted by the reduced dimensions of the display. Differing form factors therefore make for very different user experiences, with the smaller ones presumably resulting in a greater number of repetitive swipes and "tricks" to offset this, such as inertia. This represents a general problem that currently has no good solution: on a small touch screen, navigating to specific information usually requires many touch operations, due to the restricted input and presentation space. Clearly, this is an undesirable property of any user interface. Furthermore, the problem of increased input is exacerbated by the very fact that any touch operation is somewhat imprecise and tends to obscure the user interface for each interaction, since the input and display plane are the same.

Hence, the information space that needs to be accessible for presentation within typical mobile applications usually exceeds the amount of display space available. User interface design usually deals with this issue through some form of strict prioritization, presenting only what is considered most relevant to the current information need, along with the ability to navigate into other parts of the information space. Such designs therefore need input methods that allow for bringing specific parts of non-visible information into view, in a flexible and discrete manner that ideally minimizes obstruction of the display.

Looking forward, it would be reasonable to assume that any use of a tool that may alleviate these problems is highly relevant from a research perspective. One clear contender here, is the use of computer vision and pattern recognition, fields that have recently made tremendous progress. These center around how natural human language, i.e. speech, body gestures and touch can be interpreted successfully to the extent that it may either serve as the sole input for controlling a computer or work in parallel with other types of input. With increasing gains in hardware, software and falling costs, the future applicability of computer vision is currently being anticipated through active research attempts at blending frontier technology with innovative thought.

Naturally, the question arises as to whether or not this may be have any potential in mobile device interaction. For instance, what if the user's finger could be monitored, using computer vision, as it moves outside the boundaries of the display?

1.1 Contributions

The goal of the research undertaken here is to investigate a potential use of off-screen space for addressing the issues associated with small displays, as typically found on smaller portable devices. More specifically, the work explores how a touch-screen operation (a *swipe*) can extend beyond the display boundaries and into off-screen space. This leads to the question of how this space is to be defined, how the swipe transition from the display into this space will take place and how the efficiency and effect on the user experience is to be measured.

Three experiments follows a learning path that first identifies an assumed optimal spatial interface, then explores this in more detail and finally attempts to optimize it based on observed properties of the true space. The first of these shows that incorporating off-screen space as if it was a sphere leads to a slight interaction time penalty, but also eliminates the need for redundancy and inertia. Second, if users are asked to perform estimations of spatial locations, it appears that the true space does in fact align with a sphere, but also has some inherent properties that differ. Third, modifying the interface to accommodate these properties shows a slight improvement in interaction time and serves as a potential starting point for further studies.

1.2 Organization

This document has been organized as follows. Chapter 2 provides the background on current mobile devices, tracking technologies and existing work related to the topic. In chapter 3, the exact boundaries of what constitutes the goal, and what does not, is more formally defined, including the motivations and feasibility of the proposed solution. Next, assumptions and requirements of a solution that meets these goals are presented in chapter 4, including the reasoning and evolution of technology choices. Chapter 5 enumerates the sequence of theoretical steps for solving the challenges involved with deriving a full solution. A brief commentary on the actual implementation is then given in chapter 6. The data resulting from undertaking experiments with the implementation is then evaluated and interpreted in chapter 7. Lastly, thoughts and conclusions are given in the final chapter 8.

2 Background

Much of the work undertaken here will build on using technological edge and lessons learned from existing work within the field of human-computer research. For the latter, interest is primarily in the work that relates spatial awareness on displays with full or partial elimination of mechanical input devices, such as gesture-based interaction. A brief overview of relevant terms, technology and research literature will be reviewed, with the aim of providing reference points for understanding the challenges involved and how to best approach the problem.

2.1 Mobile Interaction

While clear definitions are hard to come across in the diverse and rapidly changing landscape of mobile devices, early attempts[20] at framing their interaction with humans has classified them as devices with very limited and differing resources, fluctuant communication speeds and often containing confidential information that leave them with severe security issues. For these reasons, interaction scenarios on a mobile device differ significantly from that of standard computers and most frequently centers around some communication purpose.

Usage patterns are shifting with progress however. Today, the technological trend towards minimalism could appear to be shifting mobile interaction towards *wearable computing*, understood as miniature body-worn computational and sensory devices[17]. This also changes the nature of interaction, since the devices can be discretely worn, with current examples such as *smartwatches* or wristbands that monitor health data.

2.1.1 Mobile Device Modalities

To accomplish any interaction between a human and computer, exchange of information must be provided in one way or another. Since this must necessarily take some physical form, it follows that the information must be carried through mediums that are perceptible to one or more of the human senses. Formally, these are referred to as *modalities*[2], of which sound, light or touch are the primary ones employed in today's smart devices.

For the simple case of interacting through a single modality, the interaction is referred to as *unimodal*. But an information exchange may very well be based on multiple, i.e. *multi-modal*, with the likely benefit of increasing the quality of the information exchange. A lack of modality may be problematic and cause errors, one relevant example being the difficulty of pressing virtual buttons on a touch-enabled display.

2.2 Mobile Technologies

A sprawl of different mobile device families are currently on the market. The most popular, and presumably most applicable ones, are briefly outlined here.

2.2.1 Mobile Devices

The large family of mobile devices is a rapidly evolving type of product with no strict definition, although three facts of assumed relevance will be noted here.

First, adopting a definition by the National Institute of Standards and Technology[18], a mobile device has the characteristics of a portable form factor and an operating system that is not a full-fledged operating system. Second, the most popular search engine provider, Google, recently announced[14] that mobile devices have surpassed standard computers in terms of search activity - a trend also supported by statistics collected by the world's largest major network company, Cisco[6]. Third, the most popular mobile devices, i.e. current phones and tablets[1], are all designed to run on battery and therefore have processors mainly aimed at achieving low power consumption rather than speed.

These facts together underline that mobile devices are disrupting existing consumer behavior as forerunners in their particular field, although still not computationally intended to perform beyond a limited set of well-defined and lighter tasks.

2.2.2 Hybrid Devices

Another indicator of the consumer appeal associated with mobile devices, are several attempts from major manufacturers at bridging the gap between the portability of a mobile device and the power of standard laptops. These devices try to combine the best of two worlds by incorporating faster chips and balance that choice off with extra battery power. While the hybrid definition is somewhat blurry from there on, one additional trend appears to be an ambition to combine the dual modes of minimalism and productivity by supporting both touch and keyboard input, facilitated by various novel engineering designs.

As of writing, two of the world's largest technology companies, Apple and Microsoft, each offer touch-based mobile device products, and each of these overlap into the hybrid market with corresponding "pro" versions.

[1] Such as iPhones, iPads, Android and Blackberry units.

The pro version of the former runs a restricted operation system, while the latter is based on a classic operating system, i.e. with full flexibility and freedom.

2.3 Tracking Technologies

With the recent surge in computer vision, several popular motion tracking technologies exist today, with some tailored towards specific purposes and some capable of full-flung body tracking. Two of assumed relevance will be reviewed here.

2.3.1 Full-body tracking

As an example of high-end motion tracking with six degrees of freedom, the OptiTrack system currently serves as one of the most popular and is widely cited within the academic research community. The system is essentially based on synchronizing a large number of cameras, such that the spatial positions and/or orientations of special retroreflective markers may be inferred. Under ideal conditions, and with the cameras covering a wide range of angles that combined provide the system with continuous visibility of all tracking markers, it becomes possible to perform motion tracking at sub-millimeter accuracy and at distances of up to 30 meters[19]. Hence, it is powerful in terms of its applicability, but also heavyweight and potentially over-powered, depending on the application.

2.3.2 Low-cost tracking

With a steady introduction of low-cost tracking devices in recent years, the repertoire of tools for gesture-based interaction has been growing steadily and now allows anyone with creative thought to explore a vast array of new research questions on a limited budget. In its simplest form, gestures may now be inferred by minuscule cameras alone, a strategy currently being pursued by major tech giants. For instance, one Samsung project now allows users to use their hands as interactive displays[26]. More disruptive developments include that of the Google Soli chip, which is capable of extracting human intent from close-range radar wave reflections. This then provides a 360-degree vision that perceives the physics of a hand in its entirety, as opposed to the surface alone. Researchers from Carnegie Mellon have pursued a slightly similar concept[29], using high-frequency electrical signals from a ring, emitted upon touching the opposing arm. The finger position is then inferable by comparing an intercept delay from two sensors located on a wristband. Clearly, what these new developments have in common are clever attempts at using multi-modal feedback from the human body itself, i.e. touch, sound and vision, which effectively diminishes the need for costly materials and superfluous technology.

The development of affordable tracking has also partially been market-driven by a gaming industry that has been alive and growing since the early days of the computer itself. Especially simulator games have sought to heighten the sense of realism, by additional input methods that expand the field of view beyond the capabilities of the display[2].

Another and quite different factor that has had a significant impact on the efficiency and availability of affordable tracking devices is the field of Machine Learning. Tremendous gains have been made in this field within the past two decades. As an all-data based discipline and combined with the transparency of publicized research, open-source projects have emerged that vastly improves specialized usage patterns in motion tracking and is available for anyone to use[3].

As one could expect, closed-source commercial products have too attempted to harvest the synergy of these concurrent developments. One clear contender to pave the way for low-cost tracking is the Kinect, an ambitious full-body motion tracker that was released by Microsoft in 2010 as a novel gesture-based game console controller and accessory. As somewhat of a surprise, the product was never widely adopted by the gaming community and has illuminated the fact that technological challenges still persist for such type of product[4]. Despite having the best imaginable launchpad - financial backing, a place in a healthy commercial ecosystem and the latest in motion tracking - the product still lacked commercial success. This was partly due to some requirements on the physical space in which it is used and some of its novel features have inaccuracies for which the end users had little tolerance[4].

In a surprising twist however, while slowly being abandoned by its original target audience, the Kinect is now instead making its way into the more stringent-friendly human-computer interaction research body, as a low-cost tracking device with excellent depth detection. This transition was initially made possible through various hacks that made the device work on other platforms than its intended game console. These have since evolved into more organized open-source solutions, such as openkinect.org. Realizing its hidden potential, Microsoft eventually provided windows compatibility and

[2]Such as the popular TrackIR from NaturalPoint, which achieves head tracking using retroflective markers and/or LED emitters.

[3]Such as the popular OpenCV, an open-source computer vision programming library that supports all commonly applied machine learning algorithms.

[4]Current usage of the Kinect is low and dwindling, as announced by Micrsoft in the fall of 2015[25].

an interface that now allows developers to explore the capabilities of the Kinect for custom interaction scenarios.

What mainly sets the Kinect apart from its equivalent low-cost peers, is its position as the first real-time depth camera capable of handling a wide variety of body sizes, undergoing many general motions, and run on consumer-hardware in interactive rates[21]. Furthermore, the depth detection is surprisingly good for a device that is described as fundamentally being based on triangulation[5]. However, the Kinect's depth measurements do occasionally fluctuate and the device does not necessarily obtain full depth coverage of the scene for every single frame[13]. Also, work by Khoshelham[16] show that distances do matter for the Kinect, i.e. the systematic errors of its data output do not deviate significantly from that of high-end scanners, but only for measurements obtained from within distances of one to three meters from the sensor. These close-proximity capabilities are showcased by collaborative work between Microsoft and several Universities[13], in which they reconstruct three-dimensional models in real-time by moving the sensor around a scene or object[5].

While the inner works of the Kinect is beyond the scope of this paper, briefly touching upon the basics of will help understand important details on proper use[6]. In essence, the Kinect is a light scanner that combines a number of techniques to accurately achieve its main purpose: determining the joint positions of an individual, such that body posture and gestures may be derived with high probability. Part of the secret to accomplishing this, and what makes it unique, is a patented technique of factory-encoding a known *speckle-pattern*[22] which is emitted as UV-light, such that it appears invisible to the human eye. A capture of this is shown in figure 1. This proves to be advantageous because since each speckle is unique, the subsequent retrieval of their individual deflection is identifiable and gives clear hints about the reflected surface, as well as distance[7]. Having obtained these data, body pixels are extracted and body parts classified using a Random Forest, in a process publicized by Microsoft Research[21]. From their work, it is also seen that the learning algorithm is optimized to a few hundred sequences of body poses typical for game scenarios. What is also mentioned is that the Random Forest has been trained on single finger tracking (for menu navigation) and is believed to have room for handling unseen poses.

To sum up, ideal use of the Kinect is best achieved by minimization of external UV-light, using postures that face the sensor to some extent and provide a full-body view in relatively close proximity to the unit.

[5]Enabling dream scenarios such as obliterating porcelain plates or graffiti painting, in the comfort of the home

[6]The Kinect's use of structured light is to some extent available through the original patent applications, but the format of these do not support higher-level understanding very well - the comprehension given here is therefore based on a second-hand account in the form of university material[12] that clearly labels the original patents as the sources

[7]In more detail, these attributes are inferred using the *parallax* effect, the degree of blur (depth from focus) and the use of astigmatic lenses in perpendicular dimensions for exploiting increasing skew distortion as a function of distance

Figure 1: The UV-light emitted by the Kinect, tainting everything in its path with a speckle pattern. Processing this reflection then reveals the scene in great detail.

2.4 Related Work

The concept of interacting with a computer through gestures in space has been explored for many years and a plethora of existing research already exists. We will take into account conclusions from these that relate mostly to our task, i.e. the problems and challenges of interacting in upwards of three dimensions on smaller displays using spatial interactions.

2.4.1 Spatial awareness on mobile devices

For obvious reasons, the small viewport of a portable device severely reduces the number of visual references and may confuse the viewer as to where he or she is currently located. For this same reason, sensing information of interest that reside outside the viewport also becomes difficult. In other words, maintaining spatial awareness is a difficult on smaller displays.

Researchers have pursued ways to overcome this, such as using small visual cues for providing the user with subtle references of off-screen information. Examples include work by Baudisch and Rosenholtz[3] which shows that, for data that may be visualized in two dimensions, the use of geometric marker scaling for points-of-interest (and residing out of view) leads to a noticeable speed optimization. For three-dimensions however, there is currently a lack of effective techniques for visualizing off-screen information, as has been concluded in similar work by Jäckle et al.[15].

An early and radically different strategy, as exemplified through work by Yee[28], keeps the input space

fixed in relation to the physical surroundings and lets the user develop his or her awareness by moving the viewport around in space - thereby "peeping" into areas of interest. While enticing in its concept, the approach has not been widely adopted, except for special use cases (such as star gazing applications). Furthermore, Yee concluded that the technique is physically tiring for anything but short-term interactions and suffers from (at the time, in 2003) poor sensor accuracy.

What may be concluded here, is that mobile interaction is most efficient when using a two-dimensional user interface, with some elements to support spatial awareness for the current location and possibly other areas of the information space that reside out of view.

2.4.2 Using imaginary extension planes

With the ongoing efforts at exploiting sensor technology, researchers are actively trying to predict future use of the space surrounding mobile devices. Many ideas are based on the obvious - expanding the input space by imaginary planes that extend and align with the display plane itself.

One such example is an idea by Hasan et. al[10] that bins the off-screen space in a radial fashion and successfully accelerates certain tasks, such as list browsing. Furthermore, they found the optimal expansion radius to be within 40 centimeters from the device and that optimal interaction occurred on the side of the dominant hand.

Similar work by the same author[11] compares map navigation using traditional *flick-and-pinch* touch-input to that of navigating by off-screen input instead - i.e. anchoring to the (projected) index finger and onto an imaginary extension of the display. What was discovered here, was that for basic navigation, the off-screen interaction showed improved performance in terms of both a decrease in navigation time and number of redundant touch inputs (also referred to as *clutches*).

2.4.3 Using spatial depth

While the ability to interact through space brings the additional dimension of depth into play, its applicability in three-dimensional navigation could appear to be somewhat constrained. It has e.g. been shown by Bhuiyan and Picking[7] that using the arm and hand to make selections by pointing at a large display may be fast, but inaccurate. Hence, confidently making selections by moving the hand in a three-dimensional space was in fact slow, in addition to being physically demanding. Navigating a two-dimensional plane however, was optimal in terms of providing good accuracy and speed, with test subjects able to make precise selections rapidly (and explicitly expressing their favor with the two-dimensional approach).

2.4.4 Optimal use of depth

Since depth is hampered by inaccuracy, one possible strategy could be to simplify it by discretization, in style similar to the aforementioned binning of imaginary planes. Such an approach improves the time taken to perform some motion, easing on the need for accuracy with loss of a continuous input range as a result. Such an idea has many real-world analogues for interaction scenarios where time is crucial (such as the division of the neck of a musical string instrument into frets).

One extreme case of binning, binary, would be the obvious choice for scenarios such as "clicking" an imaginary button in mid-air. Of interest here is work by Vogel and Balakrishnan [24], in which they explore applying a third dimension through the concept on an *AirTap*. This tapping of an imaginary button is shown easy to achieve by the simple exceeding of certain thresholds for relative positioning and acceleration. An identified challenge in their work is the inherent problem with ambiguity when tapping in mid-air, as many hand movements may resemble the definition of an air tap. Also, the same set of hand gestures tend to vary slightly for different individuals. This further underlines the difficulty of depth, even for the simplest of cases.

3 Research Questions

This section seeks to narrow in on initial and higher-level considerations, such as defining relevant research questions, goals, motivations and whether a solution is at all feasible given the current technological landscape.

3.1 Questions and Hypothesis

In order to swipe outside the boundaries of a mobile display, we seek to address the following:

- Are there any obstacles present in attempting to use tracking technology on mobile devices?
- Given the ability to track a user's hand in real-time, how can that information may be exploited to let the user swipe beyond the display boundaries?
- How can the off-screen space be defined, such that the transition between spaces appear continuous and accurately translates the user's perception of location into efficient navigation of the information space?

From these questions, we pose the following hypothesis:

> *Current mobile technology allows for touch-input interactions to extend outside the boundaries of the display, in such a way that the overall user experience is improved.*

This hypothesis will either be confirmed or falsified, through the concept's design, implementation, experiment and subsequent evaluation.

3.2 Goals and non-goals

To address our questions, our goals are to

- Illuminate the potential and applicability of the Air-Swipe concept, including possible drawbacks
- Formulate and implement a solution using theoretical methods in a scientific manner and with appropriate choice of technology
- Measure the impact of the solution through a test setup that reflects expected usage patterns of off-screen space

In contrast, the following is not part of our aim:

- The implementation of a solution that is applicable to any environment, such as the wide range of lighting conditions that human activity takes place within
- Efficiency optimizations, such as attempts at minimizing processor or memory usage

3.3 Motivation

Pursuing solutions for incorporating hand-tracking in proximity of mobile displays is motivated by current limitations, recent technological advancements and the overall potential for advancement of human-computer interaction.

3.3.1 Advantages of off-screen space

With current input highly constrained by ever decreasing display sizes, swiping off-screen would greatly expand the input range (which will then equal the motion range of the arm instead). This represents a vast amplification of the input space and would allow for navigating applications with much larger information spaces, as well as avoiding obscuring of the display. Given the wide applicability of mobile devices, more complex applications would be made feasible and entirely new usage patterns could emerge.

3.3.2 Applications

Since most applications with significant information space currently suffer from the confinements of input on a mobile device, possible applications of off-screen input are easily identifiable:

- **Map navigation** services are currently expected to cover the entire globe. Such an immense space can only be navigated by zooming, which exploits that rapid change of height preserves map topology and therefore does not disturb the user's spatial orientation. Swiping off-screen would likely either diminish the need for zoom operations or allow them to be performed concurrently with the navigation, allowing for a faster and more continuous interaction.
- **Paging** is the ubiquitous pattern of discretizing the information space into a vast number of separate user interfaces, often referred to as *pages*. As an example, interacting with the settings of a mobile device is typically performed through an extensive number of interfaces and in hierarchies that can appear confusing. Another special case is that of electronic documents (PDFs), which are in some sense linked-lists of upwards of

thousands of interfaces. One current interface example[8] seeks to incorporate map-style zooming, using a touch gesture to transition to a zoomed-out view with the entire document is laid out in a grid fashion. Hence, applications with a large and discretized information space may very well benefit from off-screen interactions, possibly through adopting map-style navigation mechanisms.

- **Input precision** is currently a problem with touch, due to the lack of precision pointing. However, this may be overcome by the use of off-screen space. As an imaginary example, positioning a slider (a common user interface control) could be performed with greatly increased sensitivity in off-screen space, e.g. by the distance of the hand from the display.

3.4 Feasibility

With current mobile devices being based on 64-bit multi-core processors running speeds upwards of 2 GHz or more, these have more than enough processing power for performing substantial data operations in real-time. However, due to the limitations of battery technology, mobile device chips are not intended for continuous bursts, but rather tailored for low-power usage by incorporating specialized chips dedicated for common tasks[9]. Also, memory is substantial but not abundant on these devices and the operation system they run is typically characterized by strict enforcement of memory limits, forcefully eliminating applications whenever memory availability drops below a certain threshold.

As for spatial tracking, close-proximity hand-tracking of the enveloping space is not yet part of any standard mobile device, but recent developments indicate they will be. Current tracking technology is available as separate devices that rely heavily on software, data ports, substantial bandwidth and continuous processing power. Hence, hybrid devices are likely to be the preferred choice for any research scenario.

[8] Microsoft's Reader on windows 8.1.

[9] For instance, the current Apple M9 co-processor continuously collects data from integrated sensors for later processing by the main processor.

4 Approach

Here, assumptions and requirements for a possible solution are enumerated, as well as the reasoning behind the choice of technologies.

4.1 Assumptions

The solution will be shaped by certain assumptions on the technology and future usage of swiping outside the screen boundaries. These are:

- Off-screen swiping takes place in close proximity to the display, as defined by the general motion range of the human arm. With past work showing 40 centimeters as a reasonable radius from the device[10], we will consider slightly more, 50 centimeters, as the extreme of inputs
- Users interact with mobile device displays in a 10 to 30 degree downward viewing angle and distance equal to holding the device with a slightly bent arm
- The device, tracker and software to interact with are capable of retrieving and processing tracking points in real-time, defined as at least 24 frames per second
- The user will experience some loss of input confidence in the transition to off-screen space, due to the loss of a modality (touch)

4.2 Requirements

To successfully facilitate off-screen swiping, the solution is presumed required to uphold certain properties:

- **Responsiveness** that provides the user with a sense of interacting naturally and instantly with the device
- **Accuracy** in the tracking of the index finger and the visual feedback it invokes
- **Seamless transition** from on-screen to off-screen space that appears continuous to the user

4.3 Technology choices

The Surface Pro 3 and Kinect were chosen as the technological foundation of the solution. Each comes with its own set of advantages and drawbacks, as outlined here.

4.3.1 Surface as a mobile device

To get underway with the technological foundation of the solution, an appropriate mobile device with support for touch-input had to be selected. Recognizing that existing tracker technologies involve substantial processor usage and with optimizations beyond the scope, the Surface Pro 3 hybrid was the preferred choice.

Advantages

The Surface Pro provides plenty of processing power, memory and large bandwidth by way its USB-3 connectivity. In addition, it runs a highly restricted and touch-based user interface (WinRT) on top of a normal Windows 8.1 operating system. Switching to the underlying OS is both possible and intended, to support a *dual-mode* usage pattern of serving as both a simple on-the-go device and, if needed, a flexible and more complex workhorse. Hence, it doubles as both a development machine and deployment device for touch-based applications.

Disadvantages

From the standpoint of our attempt at a solution, the main problem with the Surface Pro is that the display is significantly greater than that of an average mobile device. This is assumed to be of minor relevance however, if some appropriate technique for using only a minor portion of the display is available.

4.3.2 Kinect as motion tracker

Testing out the Kinect under optimal conditions revealed a particular setup where the tracking was stable, fast and adequate for the task. Hence, it was chosen to proceed with this as the tracking device.

Advantages

The Kinect is cabled directly and offloads the computational heavy derivation of body joints to a dedicated on-board GPU. As such, the connected device receives motion capture data instantaneously, but is spared the blunt of the computational load involved.

Disadvantages

Official usage directions impose some restrictions on the lighting conditions and the angle between the tracker and individual to track. Since the learning algorithm was trained on commonly occurring body shapes and gaming playing positions, the device will hypothetically

be somewhat impaired by individuals or actions that do not match the training data very well. Also, the amount of data fed from the Kinect is enormous and is the reason for the USB-3 requirement. No lag was seen in the tracking however.

4.4 Evolution of the approach

In the early stages of surveying the technological topography, the naive idea was to use an Android smartphone, since this represents a popular and current mobile device. This choice was set back by both observed speed and connectivity concerns. First, simulating data operations in a test application appeared to be quite slow. Second, there was no way to interface with the Kinect without the extra lag of an additional proxy device. Third, the OptiTrack manufacturer does not provide an API for parsing motion capture network broadcasts on Android (Java). An idea of porting the OptiTrack c++ API to Java was abandoned, as it grew beyond trivial and appeared likely to conflict with the time frame of the project.

Instead, a windows mobile device (Nokia Lumia) was chosen to test with the OptiTrack API. This approach was fruitless for two reasons. First, the OptiTrack API does not support the particular type of processor found in the Lumia series (ARM). Second, the manufacturer of the Lumia's operating system, Microsoft, have built in (and used) a function that remotely disables the opportunity for deploying applications to any phone below the Windows Phone 8.1 version number. This became apparent during attempts at deploying a test application, as it was prevented by the block.

The final choice of device provided a good amount of resources that greatly reduced the risk of further setbacks for reasons not relevant to the end goals. However, testing the OptiTrack with the Surface revealed another problem: only a fraction of the tracking points (which are broadcasted over a LAN) actually reached the device, making the tracking data more or less useless. The root cause of this issue was never uncovered, but is presumed to be network related, possibly due to newly introduced security measures in the WinRT runtime. The OptiTrack was therefore finally abandoned in favor of the Kinect, which worked flawlessly.

5 Solution

Here, the theoretical foundation of the solution is outlined in three steps. The first step involves finding the correlation between on-screen and off-screen space, which will allow for converting any motion tracking point to an equivalent position in the information space. Such translation obviously depends on the definition of off-screen space, i.e. the shape of the imaginary extension of the display plane. This constitutes the second task, defining potential shapes of the off-screen space. Finally, in order to align with the goals of the solution it will be necessary to ensure that the transition between the two spaces is smooth and appears close to transparent to the user.

Overall, the most challenging task was found to be the first of the three, i.e. determining the exact location, orientation and scale of the on-screen space in off-screen space. The remaining two tasks were simpler and more isolated by nature, as well more comprehensible because they had incremental effects that could easily be visualized.

5.1 Correlating spaces

We will refer to the correlating of on-screen and off-screen space as the *calibration* portion of the solution. This may be broken down into well-defined sub-steps. First, the dimensions of the on-screen space (which takes the shape of a rectangle) is defined. This is required because only a portion of the Surface screen is used (for emulating a smaller display). Next, a close approximation of the spatial (off-screen) location, scale and orientation of this rectangle is determined through tracking data. Lastly, the approximation is adjusted to compensate for minor deviations between the side ratios of the two (tracking noise being one cause of this). The end result is then two identically shaped but differently sized rectangles, which implicitly give the scaling ratio between the on-screen and off-screen space.

5.1.1 On-screen input space

Our initial subtask is to derive the rectangle that represents the on-screen input space. The lines of this rectangle then marks the transition boundaries into off-screen space. Put differently, when the user initiates a swipe on-screen and moves beyond these transition lines, he or she will be navigating in the off-screen space. Furthermore, we would like this rectangle to have lines running parallel with the device form factor.

To achieve this, data points for the boundaries are collected by having the user run the index finger along the lines on the input space in a single operation, after which the input is processed. Naturally, manual input is unlikely to produce a perfect rectangle, nor have lines that run perfectly parallel with the form factor of the display. To overcome this, a fitting equal to maximizing a rectangle within these boundaries will be assumed preferable. An exaggerated example is shown in figure 2.

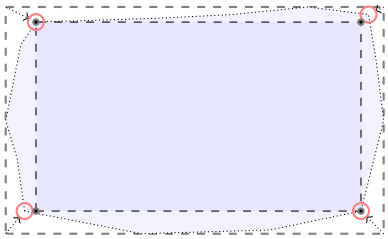


Figure 2: Fitting the input space: from the maximum device-aligned rectangle span of the input points (outer rectangle), inner rectangle corner candidates are identified (red circles) and the final fit (inner rectangle) equals the maximization of a rectangle within the space spanned by these candidates.

To produce this rectangle fit, the maximum top, right, bottom and left components are first identified from the full set of input points. This initially identifies both the location and the maximum possible dimensions of the input, corresponding to the outer rectangle in figure 2. To reach the desired smaller fit, a more appropriate set of corners is identified by, for each corner, picking the nearest neighbor from the full set of input points, i.e. the input point with shortest distance. The input space is then defined by the maximization of a rectangle within the space spanned by these new corner points.

We note that the orientation of this fitted rectangle is implicit. That is, the rectangle has sides top, bottom, left and right, as seen from the viewpoint of the user that provided the input. We will adopt these names to describe orientations, which will be relevant later on.

5.1.2 Off-screen input plane

The derivation of the device plane in off-screen space will be performed by the fitting of a set of spatial points, each assumed to be in close proximity to the plane spanned by the device. These points are in fact already available to us: in the previous derivation of the on-screen rectangle, pairings between on-screen and off-screen points were done. Hence, the fitting data is implicitly available as a set of spatial tracking data points

that should align closely with some plane (again, not perfectly due to factors such as tracking noise).

Geometrical view of the fitting

To fit a plane to the set of spatial points, we will approach this as an optimization problem with some error to minimize. An intuitive choice would be attempting to express the sum of all projection distances to the plane of choice and then use calculus to differentiate and solve for the minimum. However, this would involve calculating Euclidean distances in multi-dimensional space, which cannot easily be differentiated.

To identify an alternative fitting strategy, we will give some thought to the geometry of the tracking points. In particular, if we assume that the tracking noise is Gaussian with the mean spanning a plane, we get geometry such as that shown in figure 3.

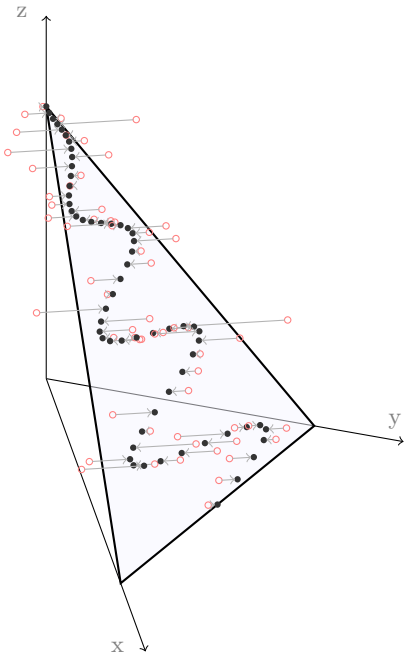


Figure 3: A sequence of spatial points, generated by departing a sine curve along the the normal of the easily comprehensible example plane $x+y+z=3$. The departure magnitude is defined by sampling from a Gaussian distribution with zero mean. Given a set of these points, this plane could therefore be considered an appropriate fitting.

Additional insight is obtained when viewing this example in profile, as shown in figure 4. From this, we see that with Gaussian noise we will see equal fluctuations on either side of the plane fit. In addition, choosing to project along a single axis should not change the plane fit, since the result is merely a scaling of the projection lengths, as illustrated in figure 5. The geometric relations of this observation are further illuminated in figure 6, which shows that this approach scales each plane-orthogonal projection length by the inverse of the angle between the plane's normal and the projection axis (i.e. a constant factor).

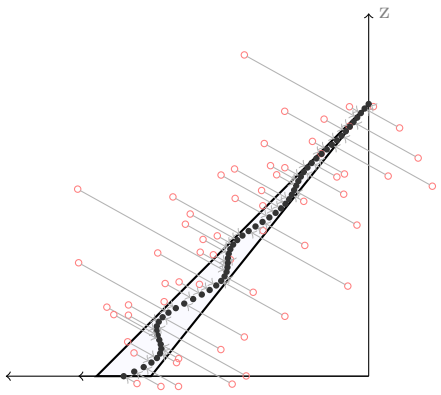


Figure 4: A profile view of the example plane with orthogonal projections.

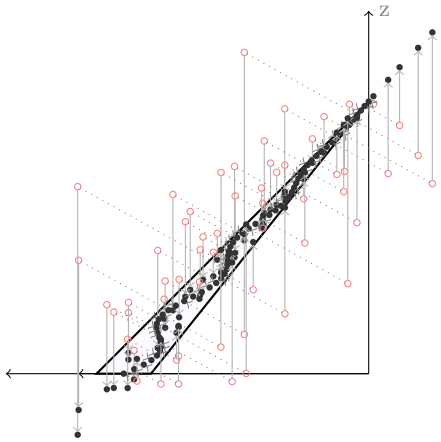


Figure 5: Projecting points to the plane along the z-axis, with previously shown projections normal to the plane shown as dotted lines. With a 45 degree plane, each projection length is now $1/sin(45°) = \sqrt{2}$ that of the orthogonal projection. While the z-aligned projections obviously obfuscate the original sine curve, the plane fitting remains unaffected, which is our sole interest.

Overall, choosing the axis-aligned projection allows for effectively eliminating the complexity of multiple dimensions, at the cost of some information loss. But the loss is irrelevant, since we only care about the ratio between the aggregation of projection distances on either side of the plane, which remain unaffected. Hence,

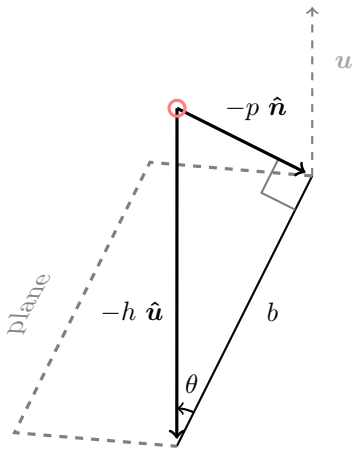


Figure 6: A closer examination of the alternative (axis-aligned) projection: given plane-orthogonal projection length p, the alternative axis-aligned projection length h will be equal to the hypotenuse of the right triangle formed by p and the planar distance b between the two projections. Hence, $h = p/\sin\theta$ (since $\sin\theta = p/h$), which means that the axis-aligned projection scales p by $\infty > (\sin\theta)^{-1} >= 1$.

under the assumption of Gaussian noise, this projection method is therefore appropriate for performing the plane fit.

Numerical precision for extreme angles

Since $\theta \approx 0$ is theoretically possible for planes close to parallel with the projection axis, that would result in large projection lengths since

$$\lim_{\theta \to 0} h = \frac{p}{\sin\theta} = \infty$$

. It would be natural to wonder if this could have any implications on numerical precision. While such conditions must necessarily increase the projection distances greatly, is should be irrelevant since we are solving for the zero-gradient of the derivative, rather than searching the solution space directly. Hence, we do not need to worry.

Fitting the off-screen plane

With a fitting strategy and confidence in the method in place, we may now proceed by finding an expression for the derivative of the aggregated projection lengths, which we will choose to perform along the z axis. Starting off, we observe that given the scalar equation for a plane

$$ax + by + cz = d$$

, we can express any single component as a linear combination of the other two. For instance, we could express the z-component as a function of the x- and y-component

$$z = \left(\frac{-a}{c}\right)x + \left(\frac{-b}{c}\right)y + \frac{d}{c}$$

, or more simply

$$z = \alpha x + \beta y + \delta$$

to lighten the notation. Our plane fitting error will therefore be defined as the sum of squared deviations

$$E(\alpha, \beta, \delta) = \sum_i \left(z_i - (\alpha x_i + \beta y_i + \delta)\right)^2$$

in which it is straightforward to see that each summation element will have partial derivatives

$$\begin{aligned}
\frac{\partial E}{\partial \alpha} &= 2x(\alpha x + \beta y + \delta - z) \\
\frac{\partial E}{\partial \beta} &= 2y(ax + by + c - z) \\
\frac{\partial E}{\partial \delta} &= 2(\alpha x + \beta y + \delta - z)
\end{aligned}$$

that are all linear in the input variables. Setting these to zero

$$\begin{aligned}
\frac{\partial E}{\partial \alpha} = 0 \ &\Leftrightarrow\ 2x(\alpha x + \beta y + \delta - z) = 0 \\
&\Leftrightarrow\ \alpha x^2 + \beta xy + x\delta \quad = xz
\end{aligned}$$

$$\frac{\partial E}{\partial \beta} = 0 \ \Leftrightarrow\ \alpha x + \beta y^2 + \delta y = yz$$

$$\frac{\partial E}{\partial \delta} = 0 \ \Leftrightarrow\ \alpha x + \beta y + \delta = z$$

makes it clear that the minimum is obtained through a linear system of three equations with three unknowns. Thus, we may sum up the zero derivative of the error in the compact matrix form

$$\begin{gathered}
AX = b \\
\Updownarrow \\
\left(\sum_i \begin{bmatrix} x^2 & xy & x \\ xy & y^2 & y \\ x & y & 1 \end{bmatrix}\right) \begin{bmatrix} \alpha \\ \beta \\ \delta \end{bmatrix} = \sum_i \begin{bmatrix} xz \\ yz \\ z \end{bmatrix}
\end{gathered}$$

which has the simple solution

$$X = A^{-1}b$$
$$\Updownarrow$$
$$\begin{bmatrix}\alpha\\ \beta\\ \delta\end{bmatrix} = \left(\sum_i \begin{bmatrix}x^2 & xy & x\\ xy & y^2 & y\\ x & y & 1\end{bmatrix}\right)^{-1} \sum_i \begin{bmatrix}xz\\ yz\\ z\end{bmatrix}$$

provided that the matrix A is invertible. To argue that this will be the case, we note that A will surely not have full rank in the case of a pure linear relationship such as $x = y$ (as clearly seen from the first two rows of A). However, the geometric interpretation of this, a straight line formed by the x and y components of the tracking points, allows us to quickly dismiss this as a special case that is unlikely to occur, since the inherent noise in the tracking data makes the probability of obtaining exact linear correlations infinitesimally small.

Hence, with A assumed invertible and the ability to solve directly for the solution component vector

$$[\alpha, \beta, \delta]^T$$

, our fitted plane

$$ax + by + cz = d$$

will then be described by

$$-\alpha x - \beta y + z = \delta$$

, where $c = 1$ because we were solving for z, i.e. implicitly using a coefficient of one.

5.1.3 Off-screen rectangle fitting

As with the processing of the on-screen data, we'll also need to fit a rectangle to the off-screen points, in order to eventually be able to precisely correlate the two spaces. However, the former on-screen case was trivial because the location and scale of the on-screen rectangle was implicit (through the minimum and maximum values of the horizontal and vertical input components). For the off-screen rectangle however, this may be located anywhere in the (previously derived) spatial plane fit and have any scale, orientation and side ratio. And again, the noisy tracking points are unlikely to form a perfect rectangle.

5.1.4 Estimating off-screen rectangle corners

To derive the rectangle, we'll first identify the expected location of the corners. We again exploit that we have previously identified the corners of the on-screen input space. Since each of these is paired with a tracking point, the four spatial corner locations are already available to us. We will refer to these spatial corner points as the *corner candidates*.

Although we could proceed immediately with these candidates, we will take into account that they are originate from a noisy stream of data. We should we care? Because minor inaccuracies could have a large impact on the solution, since the effects of a poorly calibrated plane will be amplified in subsequent spatial projections onto it (or so we assume). Hence, seeking a better approximation of the true locations may be justified and is easily achieved by averaging over a few neighboring frames, rather than settling on single ones.

The question is then, how many neighbors to include? An intuitive choice is to include frames that co-occur at the time the user's finger was located at the corner, making time our indicator of co-occurrence. So, rather than simply settling on the initial single corner candidates, these will be redefined by seeking out m neighboring tracked coordinates, using their individual capture times relative to the candidate corner as distance measure.

We will choose m, based on the assumption that a spatial point is received from the tracker every

$$(30\ {}^{\text{frame}}/_{\text{s}})^{-1} \approx 33.3 \text{ ms}$$

and that the input (finger) will not move significantly in the corner within 100 milliseconds. We therefore choose

$$\frac{100 \text{ ms}}{33.3 \text{ ms}} > 3 = m$$

as the number of spatial points to average over, meaning that we'll seek out $m - 1 = 2$ neighbors. Again, these points need to co-occur within a time span of 100 ms, a check that will be enforced in the implementation.

5.1.5 Deriving off-screen rectangle

Having settled on four spatial locations that combined should closely resemble a rectangle, we proceed to align them so that they do in fact correspond to the formal structure of a rectangle.

Constraints of a true rectangle

While we could easily just form lines between the chosen corner locations, the end result would not be purely rectangular, but a trapezium (i.e. a quadrilateral with no parallel). Hence, we will need to somehow relate all points concurrently, while matching them against constraints that capture the desired structure of a rectangle.

To define these constraints, we observe that rectangle lines are all related. The pair of lines that run

parallel (top and bottom) will share the normal vector $\boldsymbol{n}$, but different offsets c_T and c_B. The vertical pair of lines (left and right) are both orthogonal to the horizontal ones. Each too have individual offsets c_L and c_R, and share the perpendicular normal $\boldsymbol{n}^{\perp} = [-n_y, n_x]^T$. Hence, the quantities needed for describing the rectangle are offsets $\boldsymbol{c} = \left[c_T, c_B, c_L, c_R\right]^T$ and vector $\boldsymbol{n}$ (which is normal to the top and bottom lines).

Dimensionality reduction

Since the derived estimations of the rectangle corners should theoretically all be closely aligned with the already derived off-screen plane, we may as well simplify by projecting them directly (orthogonally) onto this plane. Hence, a dimensionality reduction will be performed to completely rid the dimension that is orthogonal to the plane, by finding a new two-dimensional coordinate system that represents the plane. This idea is exemplified in figure 7 and 8, which show two different viewpoints.

In more detail, suppose we have a set $\hat{\mathbb{P}}$ of three-dimensional points centered around some mean

$$\boldsymbol{\phi} = |\hat{\mathbb{P}}|^{-1} \sum_{\hat{\boldsymbol{p}}}^{\hat{\mathbb{P}}} \hat{\boldsymbol{p}}$$

, where we use the "hat" notation to indicate a vector of three dimensions. We then accomplish the reduction through a number of steps. First, the spatial center $\boldsymbol{\phi}$ is determined and all points are projected onto the (previously derived) plane using simple geometry. Next, a spatial unit vector $\hat{\boldsymbol{i}}$ of the new coordinate system is generated by crossing the plane normal with the unit vector of the x-axis. Since this will be orthogonal to the plane normal, it will be in the plane. A second unit vector $\hat{\boldsymbol{j}}$ is then produced by crossing the first unit vector with the plane's normal. This second unit vector will then be orthogonal to both the first unit vector and the plane normal. With two perpendicular unit vectors, any spatial point $\hat{\boldsymbol{p}}$ in the plane may then be converted to its equivalent $\boldsymbol{p}$ in new two-dimensional system by simple scalar projections. That is, $\boldsymbol{p}$ will have components

$$\boldsymbol{p} = \begin{bmatrix} \hat{\boldsymbol{i}} \cdot (\hat{\boldsymbol{p}} - \boldsymbol{\phi}) \\ \hat{\boldsymbol{j}} \cdot (\hat{\boldsymbol{p}} - \boldsymbol{\phi}) \end{bmatrix}$$

which is the dot product of each spatial unit vector with $\hat{\boldsymbol{p}} - \boldsymbol{\phi}$ (the vector going from the center $\boldsymbol{\phi}$ and to the point of interest). This process is illustrated in figure 9.

In addition, it is straightforward to convert any point $\boldsymbol{p}$ in the two-dimensional coordinate system back to the spatial location in the plane. This is easily

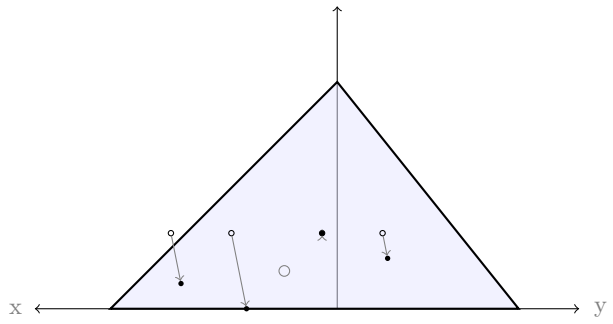


Figure 7: Exaggerated for the purpose of the example: this shows the effect of a projecting the corners of a horizontal rectangle onto a 45 degree plane in the positive octant. The mean of the projections is shown as a circle to help illuminate the geometry.

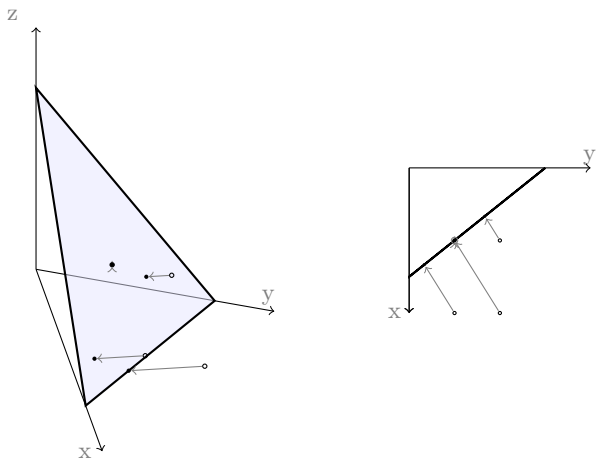


Figure 8: Additional views of the previous example, with the right figure showing a top-down view of the horizontal plane at (z-axis) height that intersects exactly with the plane (which makes it appear as a line). In practice, we do not expect such extreme fitting to occur, the simplicity of the geometry is chosen to aid the reader's comprehension of the projections.

achieved by

$$\hat{\boldsymbol{p}} = \boldsymbol{\phi} + p_x \hat{\boldsymbol{i}} + p_y \hat{\boldsymbol{j}} \tag{1}$$

, that is, scaling the unit vectors by the components of $\boldsymbol{p}$ and offset by the spatial center $\boldsymbol{\phi}$. Hence, if we are able to derive a rectangle from the points in the new two-dimensional coordinate system, we have implicitly derived a rectangle in the spatial off-screen plane, which is our goal.

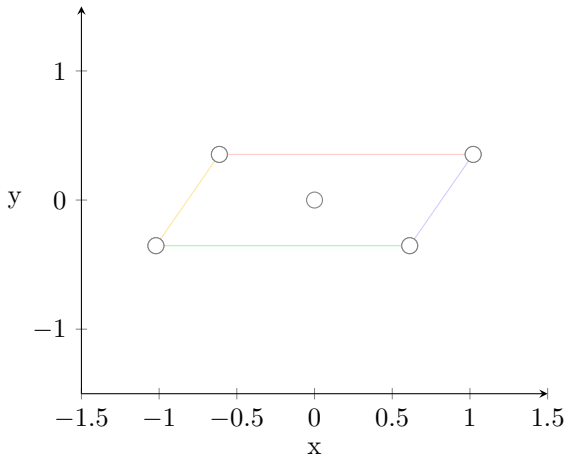


Figure 9: The result of converting the previous exaggerated example to a two-dimensional coordinate system. Fitting a rectangle to these two-dimensional points implicitly has a spatial equivalent if given two spatial unit vectors. Note a color convention has been introduced for indicating rectangle orientations (as they appear here), which will be used consistently henceforth.

Formulating the optimization problem

In broad terms, we will accomplish the fitting by attempting to minimize residuals[10]. We note that, given a point $\boldsymbol{p}$ in some plane, we may describe its distance to a line that has orthogonal vector $\boldsymbol{n}$ and offset c by the residual

$$r(\boldsymbol{p}, \boldsymbol{n}, c) = \boldsymbol{n} \cdot \boldsymbol{p} + c \tag{2}$$

. That is, the residual equals the shortest Euclidean distance to the line, which will be zero if it is exactly on the line. So, if we were to fit a line to a set of points $\mathbb{P}$, we could define the squared residual error

$$E_{\text{line}}(\boldsymbol{n}, c, \mathbb{P}) = \sum_{\boldsymbol{p}}^{\mathbb{P}} \left(r(\boldsymbol{p}, \boldsymbol{n}, c)\right)^2$$

which is minimized for some choice of $\boldsymbol{n}$ and c.

Thus, is we had four sets of points, one for each side of an approximate rectangle, we could align these to the formal definition of a (true) rectangle by minimizing the residuals for each side as

[10]In more detail, we build upon an approach for fitting parallel lines (section 6.7 of [9]), although modified for our more complex case of fitting a rectangle.

$$\begin{aligned} E_{\text{rect}}\left(\boldsymbol{n}, \boldsymbol{c}, \mathbb{P}_T, \mathbb{P}_B, \mathbb{P}_L, \mathbb{P}_R\right) &= \sum_{s}^{\{T,B,L,R\}} E_{\text{line}}(\cdots) \\ &= \sum_{s}^{\{T,B\}} \sum_{\boldsymbol{p}}^{\mathbb{P}_s} (\boldsymbol{n} \cdot \boldsymbol{p} + c_s)^2 \; + \\ &\quad \sum_{s}^{\{L,R\}} \sum_{\boldsymbol{p}}^{\mathbb{P}_s} (\boldsymbol{n}^{\perp} \cdot \boldsymbol{p} + c_s)^2 \\ &>= 0 \end{aligned}$$

using shared inputs $\boldsymbol{n}$ and $\boldsymbol{c}$, the latter being the vector of all four offsets. We note the such four sets of points in the input are already inferable, since we have previously settled on the spatial location of each off-screen corner.

Hence, the fitting of a rectangle may then be described as the minimization

$$\arg\min_{\boldsymbol{n},\boldsymbol{c}} E_{\text{rect}}(\cdots) \tag{3}$$

subject to

$$\|n\| = 1 \tag{4}$$

and the linear constraints

$$\begin{aligned} \boldsymbol{r} &= A\boldsymbol{x} \\ &\Updownarrow \\ \begin{bmatrix} \boldsymbol{r_T} \\ \boldsymbol{r_B} \\ \boldsymbol{r_L} \\ \boldsymbol{r_R} \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 0 & 0 & T \\ 0 & 1 & 0 & 0 & B \\ 0 & 0 & 1 & 0 & L \\ 0 & 0 & 0 & 1 & R \end{bmatrix} \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{n} \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 & \boldsymbol{T_x} & \boldsymbol{T_y} \\ 0 & 1 & 0 & 0 & \boldsymbol{B_x} & \boldsymbol{B_y} \\ 0 & 0 & 1 & 0 & \boldsymbol{L_y} & -\boldsymbol{L_x} \\ 0 & 0 & 0 & 1 & \boldsymbol{R_y} & -\boldsymbol{R_x} \end{bmatrix} \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{n} \end{bmatrix} \end{aligned} \tag{5}$$

that captures our parameterized definition of the rectangle structure in matrix notation. Here, we describe the tracking data through horizontally stacked matrices $\{T, B, L, R\}$, each being the vertical concatenation of an x and y column vector. Also, the vectors $\boldsymbol{r}$ are the residuals of the points in the particular rectangle side (each identified by subscript). To be explicit, if we were to expand the former we would get the more verbose

$$\begin{bmatrix} r_{T1} \\ r_{T2} \\ \vdots \\ r_{B1} \\ r_{B2} \\ \vdots \\ r_{L1} \\ r_{L2} \\ \vdots \\ r_{R1} \\ r_{R2} \\ \vdots \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & T_{1x} & T_{1y} \\ 1 & 0 & 0 & 0 & T_{2x} & T_{2y} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & 0 & 0 & B_{1x} & B_{1y} \\ 0 & 1 & 0 & 0 & B_{2x} & B_{2y} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & 0 & L_{1y} & -L_{1x} \\ 0 & 0 & 1 & 0 & L_{2y} & -L_{2x} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 1 & R_{1y} & -R_{1x} \\ 0 & 0 & 0 & 1 & R_{2y} & -R_{2x} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} c_T \\ c_B \\ c_L \\ c_R \\ n_x \\ n_y \end{bmatrix} \tag{6}$$

.

To sum up, if this optimization problem can be solved, all four lines of the rectangle will then have a known equation. Deriving the corners and center of the rectangle then becomes a trivial matter of solving for the intersections of lines and diagonals. By then, we would have derived the two-dimensional rectangle and its location in the plane. This has an easily inferred spatial equivalent through the use of (1), which will correspond to the on-screen input rectangle, as it resides in off-screen space.

Deriving the normal vector and offsets

In brief, we are able to derive the two unknowns, the normal vector $\boldsymbol{n}$ and offsets $\boldsymbol{c}$, by recognizing the problem as a special case of a linear squares problem with a quadratic constraint, for which a solution may be obtained using singular value decomposition. The proof and derivation thereof is of slightly technical nature and has therefore been moved to the appendix[9].

Finding the intersects

To finalize the rectangle derivation, all that is needed is determining the corners, i.e. the intersections between the lines. As previously mentioned, the trivial approach would be to obtain each intersect through a linear system of two equations in two unknowns. For instance, we could use the equation for a line (2) twice as

$$\begin{bmatrix} \boldsymbol{n}^T \\ \boldsymbol{n}^{\perp T} \end{bmatrix} \boldsymbol{p} + \begin{bmatrix} c_a \\ c_b \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\Updownarrow$$

$$\boldsymbol{p} = \begin{bmatrix} x \\ y \end{bmatrix} = - \begin{bmatrix} \boldsymbol{n}^T \\ \boldsymbol{n}^{\perp T} \end{bmatrix}^{-1} \begin{bmatrix} c_a \\ c_b \end{bmatrix}$$

for two perpendicular lines a and b that intersect in $\boldsymbol{p}$, with individual offsets and normals.

However, it should be possible to gracefully describe all four line intersections concurrently. Starting with the top rectangle line and proceeding in clockwise fashion we can build up the system as

$$\begin{bmatrix} \boldsymbol{T^R} \\ \boldsymbol{B^R} \\ \boldsymbol{B^L} \\ \boldsymbol{T^L} \end{bmatrix} = - \begin{bmatrix} \boldsymbol{n_T}^T & \dots & & 0,0 \\ \boldsymbol{n_R}^T & & & \\ \vdots & \boldsymbol{n_R}^T & & \\ & \boldsymbol{n_B}^T & & \\ & & \boldsymbol{n_B}^T & \\ & & \boldsymbol{n_L}^T & \vdots \\ & & & \boldsymbol{n_L}^T \\ 0,0 & & \dots & \boldsymbol{n_T}^T \end{bmatrix}^{-1} \begin{bmatrix} c_T \\ c_R \\ c_R \\ c_B \\ c_B \\ c_L \\ c_L \\ c_T \end{bmatrix}$$

$$= - \begin{bmatrix} \boldsymbol{n}^T & \dots & & 0,0 \\ \boldsymbol{n}^{\perp T} & & & \\ \vdots & \boldsymbol{n}^{\perp T} & & \\ & \boldsymbol{n}^T & & \\ & & \boldsymbol{n}^T & \\ & & \boldsymbol{n}^{\perp T} & \vdots \\ & & & \boldsymbol{n}^{\perp T} \\ 0,0 & & \dots & \boldsymbol{n}^T \end{bmatrix}^{-1} \begin{bmatrix} c_T \\ c_R \\ c_R \\ c_B \\ c_B \\ c_L \\ c_L \\ c_T \end{bmatrix}$$

where we have adopted the shorthand symbols

$$\{\boldsymbol{T^R}, \boldsymbol{B^R}, \boldsymbol{B^L}, \boldsymbol{T^L}\}$$

for compactly describing the $[x, y]^T$ intersects of two rectangle sides, i.e. top-right, bottom-right, bottom-left and top-left. In the notation, we also exploited the orthogonality of the shared normal $\boldsymbol{n}$ for the two dimensions (top and bottom versus left and right).

Shuffling the rows of this system around

$$\dots = - \begin{bmatrix} \boldsymbol{n}^T & \dots & & 0,0 \\ \boldsymbol{n}^{\perp T} & & & \\ \vdots & \boldsymbol{n}^{\perp T} & & \\ & \boldsymbol{n}^T & & \\ & & \boldsymbol{n}^T & \\ & & \boldsymbol{n}^{\perp T} & \vdots \\ & & & \boldsymbol{n}^{\perp T} \\ 0,0 & & \dots & \boldsymbol{n}^T \end{bmatrix}^{-1} \begin{bmatrix} c_T \\ c_R \\ c_R \\ c_B \\ c_B \\ c_L \\ c_L \\ c_T \end{bmatrix}$$

$$\Downarrow$$

$$
\begin{aligned}
&- \begin{bmatrix}
\boldsymbol{n}^T & \dots & & 0,0 \\
\vdots & \boldsymbol{n}^T & & \\
 & & \boldsymbol{n}^T & \\
 & & & \boldsymbol{n}^T \\
\boldsymbol{n}^{\perp T} & & & \\
 & \boldsymbol{n}^{\perp T} & & \\
 & & \boldsymbol{n}^{\perp T} & \vdots \\
0,0 & & \dots & \boldsymbol{n}^{\perp T}
\end{bmatrix}^{-1}
\begin{bmatrix} c_T \\ c_B \\ c_B \\ c_T \\ c_R \\ c_T \\ c_L \\ c_L \end{bmatrix} \\
&= - \begin{bmatrix}
n_x & n_y & & & & & \dots & 0 \\
0 & & n_x & n_y & & & & \vdots \\
\vdots & & & & n_x & n_y & & \\
 & & & & & & n_x & n_y \\
-n_y & n_x & & & & & & 0 \\
 & & -n_y & n_x & & & & \vdots \\
\vdots & & & & -n_y & n_x & & \\
0 & \dots & & & & & -n_y & n_x
\end{bmatrix}^{-1}
\begin{bmatrix} c_T \\ c_B \\ c_B \\ c_T \\ c_R \\ c_T \\ c_L \\ c_L \end{bmatrix} \\
&= - \mathcal{N}^{-1} \boldsymbol{c} \\
&= - \mathcal{N}^T \boldsymbol{c}
\end{aligned}
$$

and we reach a structure, from which it is clearly seen that the rows and columns in fact represent orthogonal unit vectors. Hence, with $\boldsymbol{n}$ being unit length, $\mathcal{N}^T\mathcal{N} = I$ must be the case since $\mathcal{N}^{-1} = \mathcal{N}^T$. All intersects may therefore be solved for directly and without matrix inversions. Knowing these intersects, we then have the location, orientation and scale of the on-screen input space, as it resides in off-screen space. To illustrate the result and corresponding spatial translation, figure 10 shows a fitting of one previous example. Figure 11 then shows different views of the corresponding spatial translation.

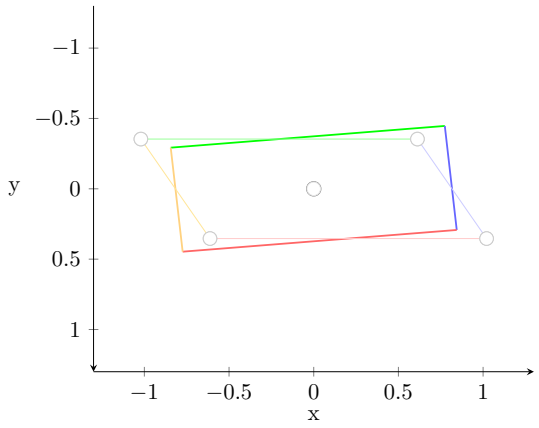


Figure 10: The result of performing a rectangle fit on the previously given example, which looks intuitively correct in both size, shape and orientation (with orientations identified by line coloring).

5.1.6 Rectangular side ratio fitting

We have now reached a point where both two-dimensional on-screen input points and their paired three-dimension tracking points can be translated into true rectangles by way of a fitting strategy. However, it is rather unlikely that the two rectangles should have the exact same side ratio, given the noise in the data. In order to correlate the two, some alignment therefore needs to be done. That is, the side ratios of the two need to match up.

We will assume that any adjustment of the on-screen space is undesirable, since it is calibrated to fit specific dimensions exactly (the rims that simulate the exact dimensions of a smaller mobile device). The on-screen space therefore determines the target ratio

$$\theta = \frac{r}{b} \geq 1 \tag{7}$$

which is the ratio between the on-screen horizontal side r and the shorter vertical side b. We will denote the pre-adjusted dimensions of the off-screen space with the corresponding τ and δ, where $\tau/\delta \neq \theta$. Put differently, given with the expectation of minor deviation between the two rectangles, we do not expect to see $r = \tau$ and/or $b = \delta$ in the strict numerical sense.

The question is therefore how to chose a meaningful strategy for performing the alignment, i.e. the adjustment of the off-screen rectangle sides. An obvious idea would be to minimize the difference between the areas of the pre- and post-fitted rectangle. Since the post-fitted area may be either smaller or larger than the pre-fit, a least squares approach

$$
\begin{aligned}
E &= (A_{postfit} - A_{prefit})^2 \\
&= (rb - \tau\ \delta)^2
\end{aligned}
$$

would be appropriate as error measure to minimize.

Disregarding the target ratio constraint θ for a moment, we may gain insight into the solution space by parameterizing one rectangle in this error definition. The derivative of the error with respect to τ and δ would be

$$
\begin{aligned}
\frac{\partial E}{\partial \tau} &= -2\delta(\tau\delta - rb) \\
&= 2\delta(rb - \tau\delta) \\
\frac{\partial E}{\partial \delta} &= -2\tau(\tau\delta - rb) \\
&= 2\tau(rb - \tau\delta)
\end{aligned}
$$

and we could then identify the minimum by setting

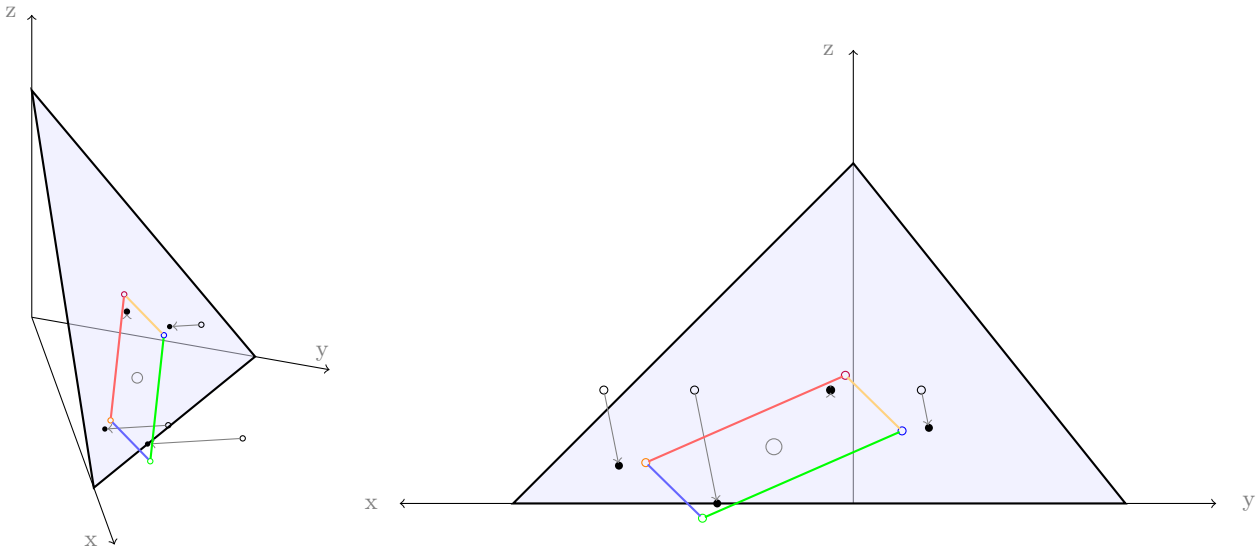


Figure 11: The spatial translation of the rectangle fitting, as seen from two different viewpoints in the positive octant.

these to zero as

$$\begin{aligned} &\arg\min_{\tau,\delta} E \\ &\Updownarrow \\ &\frac{\partial E}{\partial \tau} = 0, \;\; \frac{\partial E}{\partial \delta} = 0 \end{aligned}$$

. Solving the two yields

$$\begin{aligned} \frac{\partial E}{\partial \tau} = 2\delta(rb - \tau\delta) = 0 \\ \Updownarrow \\ 2\tau\delta^2 = 2\delta rb \\ \Updownarrow \\ \tau\delta = rb \\ \Updownarrow \\ 2\tau^2\delta = 2\tau rb \\ \Updownarrow \\ \frac{\partial E}{\partial \delta} = 2\tau(rb - \tau\delta) = 0 \end{aligned} \tag{8}$$

i.e. an identical solution for both derivatives. Hence, the error is minimized when the pre- and post-fit rectangles have the exact same area $\tau\delta = rb$, but with potentially differing ratios. This seems intuitively sensible. With this goal in mind, returning to (7) we see that

$$\begin{aligned} &\frac{\tau}{\delta} = \theta \\ &\Updownarrow \\ &\tau\delta = \delta^2\theta \end{aligned}$$

which, combined with our insight from (8) that minimization of error is obtained by equating the areas, gives

$$\begin{aligned} &\tau\delta = rb \\ &\Updownarrow \\ &\delta^2\theta = rb \\ &\Updownarrow \\ &\delta = \sqrt{\frac{rb}{\theta}} \end{aligned}$$

which has only the positive solution. Using (7) again, we get

$$\begin{aligned} \tau &= \theta\delta \\ &= \theta\sqrt{\frac{rb}{\theta}} \\ &= \sqrt{\theta rb} \end{aligned}$$

which leaves us with the solution for performing the adjustment with minimum error. To briefly verify, it is trivial to confirm that for the special case of $r = b = l$ (a square of side length l) we would get $\delta = {}^{l}/\sqrt{\theta}$ and $\tau = l\sqrt{\theta}$, such that ${}^{\tau}/_{\delta} = \theta$, which matches our definition of the target side ratio.

All that is left is determining how this rescaling of the off-screen rectangle affects its position. The simple and intuitive choice would be to anchor the rectangle on its center and that is the approach taken here.

5.2 Off-screen Space Definitions

With the rectangular on-screen space defined and the location, orientation and scale of its spatial equivalent derived, the question is then how to interpret the incoming

screen motion capture data. That is, how the "shape" of the off-screen space is to be interpreted, which determines how the user interfaces with the information space. While the small on-screen space is planar by physical construction, the same need not apply to users' perception of the invisible off-screen space. Three intuitive ideas will be pursued:

- **Planar-orthogonal** defines the virtual extension of the display as planar and projects spatial points orthogonally onto that plane
- **Planar-directional** also employs the planar extension, but performs the plane projection along the direction of the finger
- **Spherical** curves as if the user was inside sphere and projects along the direction from shoulder, through finger tip and onto the spherical boundary

The first of these interfaces, planar-orthogonal, is straight-forward to comprehend. For the second, planar-directional, the direction of the projection is defined as going from base of the finger and to its tip. The third, the spherical definition, obviously depends on the location of the sphere center and its radius. The idea for this interface came from the observation that the natural motion of the arm follows a somewhat spherical curve. Hence, the sensible choice would be to anchor such sphere at the shoulder and define its radius as the distance to the imaginary plane spanned by the device. While this sphere definition is clearly dynamic (as being dependent on the current location of the shoulder), it is assumed to perform less so in practice, since the user is expected to remain relatively still during the interaction.

5.3 Spatial Transition and Stabilization

With a calibrated solution and some choice of spatial interface in place, the Air-Swipe can now be achieved by switching the navigation mechanism to the off-screen interface when the user swipes outside the on-screen space. This should appear continuous and seamless to the user, as is part of our goals and requirements. However, bluntly switching between the spaces would not fulfill these very well, since their input data have differing levels of noise.

5.3.1 The need for stabilization

Both touch input and tracking data have inherent inaccuracies, but they differ in magnitude. Any data from motion tracking technology is obviously noisy, although whether it is noticeable or relevant depends on how it is applied. On the other hand, the inherent noise in on-screen touch input is pre-processed by a touch-controller, which mitigate errors incurred by electrical noise, scaling factors and mechanical misalignments[8]. Hence, on-screen input is automatically stabilized by the device, whilst off-screen input is not.

With regards to the application here, this difference will be visible when transitioning to off-screen space as negligible but distracting "jitter" fluctuations around the translated position. As previously accounted for, the Kinect tracker does in fact leave the client to deal with noise handling[13]. Hence, a more continuous and user-friendly translation may be achieved by applying similar pre-processing of the tracking data.

The smoothing strategy pursued here was to achieve translations that appear smooth to the human vision, but still aligns with the requirement of responsive input. In the geometric sense, we will therefore attempt to stabilize by "pulling" new motion tracking points in the direction of the immediate past, but less so for higher velocities of movement. In the following, we first define the general incorporation of past observations and next, how this is used in conjunction with the variables of time and velocity.

5.3.2 Stabilization method

To achieve a weighting between the present and past, we choose to define the smoothed off-screen location $\boldsymbol{\tau}$ as

$$\begin{aligned} \boldsymbol{\tau_{n+1}} &= \boldsymbol{t_n} - \alpha_n \boldsymbol{\delta} \\ &= \boldsymbol{t_n} - \alpha_n(\boldsymbol{t_n} - \boldsymbol{\tau_n}) \\ &= \alpha_n \boldsymbol{\tau_n} + (1-\alpha_n)\boldsymbol{t_n} \end{aligned} \tag{9}$$

where $\boldsymbol{t_n}$ is the n-th incoming (unprocessed) tracking point, $\boldsymbol{\tau_n}$ is the previously (processed) location and the difference $\boldsymbol{\delta}$ between the two scaled by $0 \geq \alpha \geq 1$[11]. To understand the behavior of this, observe, that the extreme example of $\alpha = 1$ will result in exclusively relying on the past, while the opposite extreme of $\alpha = 0$ evalutes to the most recent raw observation only (that is, no smoothing at all). With the goal in mind of finding some middle ground $0 < \alpha < 1$, we first confirm that the nature of the recurrence is sensible. By expanding (9) we see

$$\begin{aligned} \boldsymbol{\tau_{n+1}} = {} & (1-\alpha_n)\boldsymbol{t_n} + \\ & \alpha_n(1-\alpha_{n-1})\boldsymbol{t_{n-1}} + \cdots + \\ & (\Pi_{i=j}^{n}\alpha_i)(1-\alpha_{n-j})\boldsymbol{t_{n-j}} + \cdots + \\ & (\Pi_{i=0}^{n}\alpha_i)\boldsymbol{\tau_0} \end{aligned}$$

[11] This approach is an adaptation of a CPU-scheduling technique presented in [23], but extended to incorporate distinct weights α for each observation n (and also, with flip of symmetry in the definition of α).

which shows how the significance of the past diminishes very rapidly with increasing number of observations n, exponentially in n if all weights α are approximately equal.

5.3.3 Dynamic penalization

With the weighting strategy in place, the question is then how to define the weight α for each observation n, such that it encapsulates the desired effects of time and velocity. We will do so by defining the respective weight constituents π and ν.

Penalizing the past

With regards to time, the goal is to impact α such that the immediate past holds large weight and infinitesimally small attention is given to more distant past. This desired impact profile is shown in figure 12. An exponential decrease would be suitable to capture this, defined as

$$\pi_n = \gamma^{-t_n}$$

where t_n is time (milliseconds) since the previous $(n-1)$-nth observation and γ is some constant base to be determined by visual inspection.

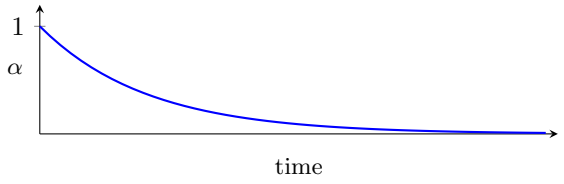


Figure 12: Desired impact profile on α as a function of time

Penalizing velocity

To also take velocity into account, we will consider the spatial length of the current movement vector $\boldsymbol{\delta}$. First, we will want to ensure a responsive interface by fully discarding the past for very fast movements, i.e. some velocity threshold, but obviously not so low so as to be triggered by the inherent velocity of the noise. Also, we would like some sharp, but smooth transition in reaching this threshold, with profile as shown in figure 13. We may capture such profile and threshold by a combined positive polynomial and cut-off function

$$\nu_n = \max\left(0, 1 - \lambda\|\boldsymbol{\delta}\|^2\right)$$

for some constant λ determined by visual inspection.

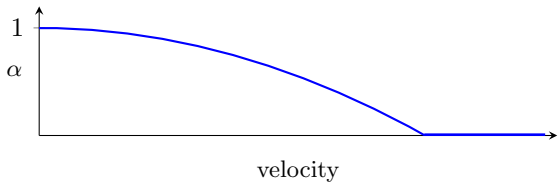


Figure 13: Desired impact profile on α as a function of velocity

Combined penalizations

Incorporating the two penalizations in (9) gives the recurrence

$$\begin{aligned}\boldsymbol{\tau_{n+1}} &= \boldsymbol{t_n} - \alpha_n\boldsymbol{\delta} \\ &= \boldsymbol{t_n} - \pi_n\nu_n\boldsymbol{\delta} \\ &= \boldsymbol{t_n} - \left(\gamma^{-t}\right)\left(\max(0, 1 - \lambda\|\boldsymbol{\delta}\|^2)\right)(\boldsymbol{t_n} - \boldsymbol{\tau_n})\end{aligned}$$

which is the algorithm for ensuring a smooth transition from on-screen to off-screen space, such that the user's sense of continuity is maximized.

To conclude, we will remark briefly on the expected behavior of this. First, the penalty imposed by time is assumed to be close to constant, since the frame rate of the tracker is approximately constant. Second, rapid navigations that exceed the velocity cut-off are assumed to be rare and constitute only a minor portion of all navigation. Hence, the general swiping speeds are not expected to fluctuate wildly, but rather increase or decrease in a relatively smooth manner. As such, the penalty imposed by velocity is also somewhat constant. This means that the weights do not fluctuate much within brief time periods, such that

$$\alpha_0 \approx \alpha_1 \approx \cdots \approx \alpha_n$$

. If so, the recurrence will then resemble

$$\begin{aligned}\boldsymbol{\tau_{n+1}} = (1-\alpha)\boldsymbol{t_n} + \\ \alpha(1-\alpha)\boldsymbol{t_{n-1}} + \cdots + \\ \alpha^j(1-\alpha)\boldsymbol{t_{n-j}} + \cdots + \\ \alpha^{n+1}\boldsymbol{\tau_0}\end{aligned}$$

which is the *exponential average* of past observations. This is the weighting between the present and past that is assumed intuitive for stabilizing the input in general, but one that still ensures responsiveness by favoring the present for large velocities.

6 Implementation

In this section we touch very briefly on design choices and lessons learned from implementing and configuring the solution.

6.1 Choice of runtime

The solution was implemented in the WinRT runtime, which runs natively on the Surface Pro. It has excellent support for developing touch-enabled user interfaces, such as those found on mobile devices. It is also a newer runtime partially derived from the widely popular .NET, but with limited and complicated backwards compatibility due to being a super-subset of this.

6.2 Mathematical routines

Some parts of the solution rely on substantial use of linear algebra for matrix operations, decompositions etc. Stable libraries already exist for this type of logic and one popular choice, the Accord.NET framework was chosen to work with. However, this entire framework needs to be re-targeted and recompiled (to Windows 8) in order for it to work in the WinRT runtime. In addition, some parts of the source code had to be manually changed, although we will omit further technical detail here.

6.3 Application design

The overall design of the application provides several functionalities, primarily for the calibration of spaces, execution of experiments, and data processing/extraction. Several interfaces also provide the ability for confirming application parameters by inspection, such as projection information (lengths, delta movements etc.) and three-dimensional rendering of the tracking data as well as visual projection pointers for determining the optimal tracker setup. Part of this is shown in figure 14.

6.4 Application parameters

As previously accounted for, some smoothing parameters needed to be determined by visual inspection. The effects of these were estimated by observing the translated off-screen interaction in the on-screen space, i.e. the projections of tracked joints of interest onto an imaginary extension of the plane spanned by the device. That its, off-screen input was visualized on-screen through the positioning of circle "cursors" in the user interface, which visually reveals any presence of noise.

For the general smoothing that incorporates the past, an optimal $\gamma = 1.0035$ was found by parameter sweeping through a value range until no jitter was visible. To reason briefly, for a frame rate of approximately 30 frames per second, this corresponds to $\pi = 1.0035^{-30} \approx 0.0573$, which is equivalent to incorporating only 1/17 of the previous frame. As such, eliminating jitter noise compromises little on the information of the most recent tracking frame.

As for the mechanism that ensures responsiveness, a value of $\lambda = 770$ was identified as optimal in similar sweeping approach. Hence, for this choice to disregard the past completely ($\alpha = 0$), we would need to see delta movement

$$
\begin{aligned}
0 = \nu &= \max\left(0, 1 - \lambda\|\boldsymbol{\delta}\|^2\right) \\
&\Updownarrow \\
0 &\leq 1 - (770)(\|\boldsymbol{\delta}\|^2) \\
&\Updownarrow \\
\|\boldsymbol{\delta}\| &\geq 1/\sqrt{770} \\
&\approx 0.0036m
\end{aligned}
$$

, i.e. Euclidean distance of at least 3.6 millimeters for every 30 milliseconds (the aforementioned frame rate). In more comprehensible scale, this corresponds to approximately 12 centimeters per second, a number that appears reasonable as being on the border between precision and non-precision movements.

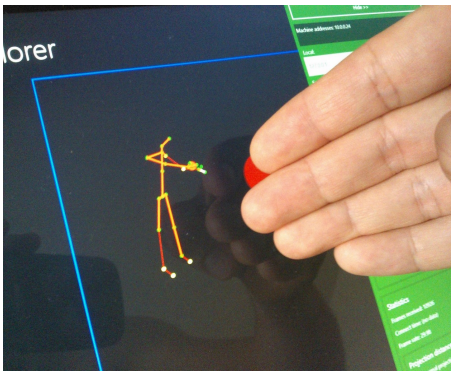


Figure 14: Shaking hands with oneself: visual rendering of the tracking data allows for exploring the stability and possible configurations of various tracker positionings. A sidebar (green on right) allows for adjusting smoothing parameters (γ and λ) and a visual cursor (partly obscured red circle in center) allows for evaluating the resulting projection behavior.

7 Results and Evaluation

To understand how well the Air-Swipe solution performs, a series of experiments were conducted, with the overall goal of providing quantifiable measurements for evaluation. Our definition of success will be the overall extent to which the solution optimizes the user experience. This will primarily be interpreted as a reduction of interaction time, but secondary factors, such as redundancy of input and sense of control, will also be taken into account.

7.1 Experimental setup

All the experiments were based on using the Surface as input device and with the smaller on-screen space artificially simulated by using rims. All tracking was done using the Kinect, which was placed in optimal conditions that were determined by experimenting with a wide variety of light conditions and tracker positions. Of all tested setups, one proved ideal in terms of providing consistently stable tracking of the body joints of interest (finger, hand and shoulder).

7.1.1 Positioning

The optimal positioning was found to be with the tracker having a profile view of the device and in equal height to it. Relative to the user, the tracker was placed on the side opposite to the interacting hand. As the user was distanced an arms-length (approx. 0.5 meter) from the device, this left the tracker with an angled forward view of the individual to track (as is required by its design). This setup, depicted in figures 15 and 16, provided stable tracking for the full range of motion, until the user's arm reached an angle that aligned it with the body profile (an extreme and awkward input position considered beyond the requirements of the experiment)[12].

7.1.2 Lighting

It was found that high volumes of both artificial and natural light had a negative impact on the tracking, which correlates with the general Kinect guideline of lower lighting conditions. Hence, only a minor degree of (non-direct) daylight was allowed into the test room and only in direction with the tracking device's point of view.

[12]For the curious reader, a video[1] of an early prototype that demonstrates this experimental setup is also available.

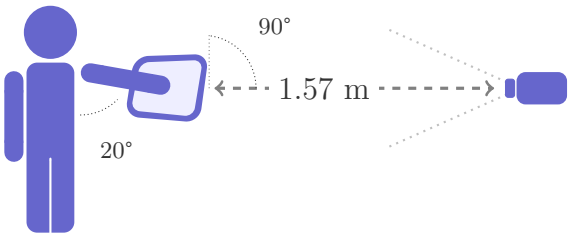


Figure 15: The experimental setup, as seen from a profile-view. This shows the participant rotated towards the viewer, for the purpose of the illustration (the actual experiment has the user standing relaxed in front of the display).

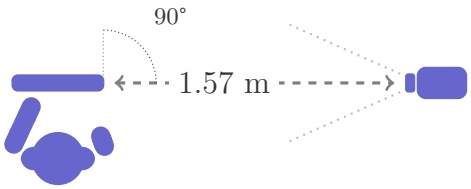


Figure 16: The experimental setup, as seen from above.

7.1.3 Simulated On-screen Dimensions

As the Surface has a significantly larger display than that of current small mobile device, a strategy for simulating the latter was necessary. Swiping outside the boundaries of a small mobile display will usually move outside the dimensions of the actual device, since the two are essentially equal. Hence, going into off-screen space will be marked by a loss of modality (touch) and introducing similar feedback will therefore approximate real-world conditions better.

The chosen approach was therefore to employ physical rims to define the smaller input space. These

Figure 17: The frame used for simulating a small input space, chosen to align with the display dimensions of a popular and small mobile device, the Samsung S6. Here shown overlaid with its close relative, the Samsung S4.

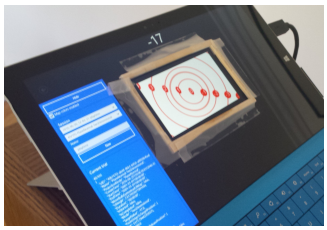


Figure 18: The simulating of the dimensions of a smaller mobile device, achieved by fixating a low-profile frame onto the display.

have some height, such that the user is given noticeable feedback in transitioning outside the boundaries. Also, this height catalyzes the off-screen interaction by "forcing" the finger into mid-air. In detail, a rectangular frame was carefully constructed, such that it had height 4mm the and the exact inner dimensions of $4.436\text{mm} \times 2.495\text{mm}$, which is representative of one currently popularized smaller mobile device. This is shown in figure 17. The frame was then fixated onto the Surface, in such a way that it aligned perfectly with the form factor of the device. The device was then ready for calibration with the implementation. This construction and its calibration is shown in figure 18.

7.2 Experiment one: spatial definitions

Initially, it was sought to explore through experimentation how different definitions of off-screen space compared against each other, including a baseline derived from using on-screen space (touch) only. The three different off-screen space definitions were chosen to test separately, as previously accounted for. For each of these, different dimensionalities were explored in isolation through variations of an appropriately designed number-panning task.

7.2.1 Tasks

The participants were asked to perform a panning task, conceived so as to be easily comprehensible and yet capable of efficiently capturing the desired metrics. Each exercise presented the participant with the simple concept of dragging a line with numbered tick marks in some numerical range that exceeded the on-screen viewport by several factors. The size of the input space was then equal to the length of this range (as visualized by the line), with an added $^3/_4$ of the on-screen viewport as "buffer" on each side. This buffer was added to ensure that navigating to an extreme of the input space still left $^1/_4$ of the line in view, thereby avoiding any loss of spatial orientation.

Each value on the line was made clearly visible as a contrast overlay on the corresponding tick mark. The task was then to drag the line until a particular trial "target value" came into center-view of a bullseye-circle (of 3.0 centimeters in diameter). This circle, initially red, turns green when the target value is positioned within its enclosing space. In order successfully complete the trial, the participant was then required to continuously hold the value within that space for 3 seconds. Each trial was fully recorded with e.g. various vector data, such as target and release positions, and durations from initial interaction until completion of the exercise.

Ideally, the task would be designed to cover a full systematic exploration in two dimensions. But due to the extent of such an experiment, this would likely fatigue the participants and decrease the quality of the data to collect. Hence, only two dimensionalities, horizontal and vertical, were included and examined in isolation. Two variations of the task were therefore designed:

- **Exercise *a*** aligns the line of numbers with the horizontal dimension and places the positive range of values on the right-hand side of the origo (under the assumption that a left-to-right increasing order is culturally appropriate for the test subject)

Figure 19: The initial viewport of the input space in exercise a (left) and b (right, as used in experiment one.

- **Exercise *b*** is as the previous, except the line is angled vertically and lists the positive numbers on the upper half (assuming the intuition of height is appropriate choice for the test subject)

The two exercises, shown in figure 19, share equal tick-mark distances and have approximately the same initial visible input range, $[-3..3]$ for task a and $[-2..2]$ for task b, which is assumed to provide an adequate sense of spatial scale. They share the same numerical tick range of $[-50..50]$ and therefore have equal dimensions of input space, which spans a distance of 1 meter. This choice of physical range is estimated to be somewhat beyond the comfortable motion range for the average user, and as such, the experiment is assumed to cover (and go slightly beyond) fringe conditions.

As previously work has shown, loss of spatial awareness is an issue on small mobile devices unless countered and may therefore also be an issue for the experimental design used here. For instance, high navigation velocities will likely be disorienting, since the only spatial references are the line numbers, which can only be read at idle or slow speeds. In addition, the user should obviously not be left without any visual references at all, as this would lead to a total loss of orientation.

These considerations were found to be very relevant, given the extreme simplicity of the task design. That is, the simplicity of the design is considered more detrimental to spatial awareness than most normal user interfaces. This was sought countered by minor design decisions.

First, it was assumed that any user interface interaction starts out with some clear reference of location. For instance, opening a text document usually starts out on the first page. Likewise, digital maps usually start out in a location familiar to the user. To align with such observations, it was assumed desirable to have the user consistently start in a well-known location, regardless of the particular trial parameters. The start location was therefore chosen to always be the origo (center) of the line, as opposed to e.g. some randomization.

Second, it was assumed that for any user interface, some indications of spatial awareness will be present during both low and high navigational velocity. To ex-

emplify, the pages of a document are distinctly different and rapidly enumerating them still provides some awareness in the form of quick glances of the unique outline of each page. This same reasoning applies to map navigation, since any top-down map view has a distinct topographical outline that is perceivable during both slow and fast location changes. Thus, to provide some awareness feedback during navigation, the tick marks were therefore made to increase in size and elliptical shape with increasing distance from the origo. Such design provides subtle hints that support spatial awareness, regardless of the navigational velocity.

In addition, a pilot test of the trials showed that users may mistake positive and negative target values. Both the target value indicator and line tick marks were therefore color coded, with negative numbers in cold color and positive numbers in warm. This color-coding is shown in figure 20 for a trial variant from one of the subsequent experiments.

All in all, the decisions made here are assumed to bring the trial design closer to the general user interface characteristics of smaller mobile devices.

7.2.2 Input interfaces

To provide a reference point for comparison, a baseline was first derived by completing the two tasks in an interface that allowed touch input only. Next, each off-screen space definition of interest was explored as a separate interface, again using the same two tasks, *a* and *b*.

Baseline

As mentioned, the baseline was derived with input confined to the on-screen space only and no use of off-screen space. Exclusive to the baseline is the incorporation of inertia, a navigational aid so common in touch-enabled interfaces that it was considered required in order for the baseline to generalize well. For this same reason, the implementation of inertia was done so as to align its behavior with what is presumed normal to most users. The inertia reference of choice was therefore that of a widely known map service provider[13]. Using on-screen inertial delta movement vector $\boldsymbol{\delta}$ and its associated velocity v (both provided by the chosen framework), the modified inertia delta movement $\hat{\boldsymbol{\delta}}$ that most closely resembles the desired behavior was found to be the exponentially decreasing

$$\hat{\boldsymbol{\delta}} = 2.7^{-(\hat{v}/v)}\boldsymbol{\delta}$$

where $\hat{v}$ is the velocity of the last non-inertial movement before the intertia was initiated. Furthermore, the magnitude of the velocity was scaled by

$$v = \min(v, 7.3)$$

to enforce an upper ceiling that avoids extreme inertia. Such a ceiling is necessary on a touch-enabled device, because excessive swipe velocities do occur frequently and applying the corresponding inertia will result in loss of spatial awareness unless capped.

Plane-orthogonal

The simplest of the off-screen definitions, this interface projects the user's finger in a line orthogonal to the plane. The user may then continue the swipe off-screen, as if an imaginary virtual extension of the display was present in the plane spanned by the device.

Plane-directional

Here, the projection follows the direction of the participants finger, defined as the vector from the base of the index finger and to the tip[14]. As the extension of the display is still planar, the velocity of navigational movements then depends not only on the orthogonal distance from the finger to the plane, but also its angle with the plane's normal.

Spherical

The most complex of the interfaces, this allows for interacting with off-screen space through projecting onto a sphere. This sphere is centered at the user's shoulder, with radius equal to the shoulder's orthogonal distance to the plane spanned by the device. This distance is assumed to correspond to the length of the user's arm, such that the finger ideally remains at the boundary of the sphere, regardless of the input position. Although this definition of radius is in fact dynamic (calculated for each received tracking frame), the user is presumed to not move around relative to the device and thus, the radius will be close to constant. However, should the user choose to move further from the device for whatever reason, the projections will take on a more directional behavior (i.e. a simple enlargement of the sphere), which is still presumed intuitive to most users.

7.2.3 Trial design

To conduct the experiment for a given interface and task, participants had to complete a set of trials, each asking for navigating to a target value unique to the

[13] Google maps

[14] In detail, some minor offset is applied through visual inspection during calibration, since the tracker provides a fix on the center of the hand (rather than the base of the index finger, as is the desired point to approximate for deriving pointing direction).

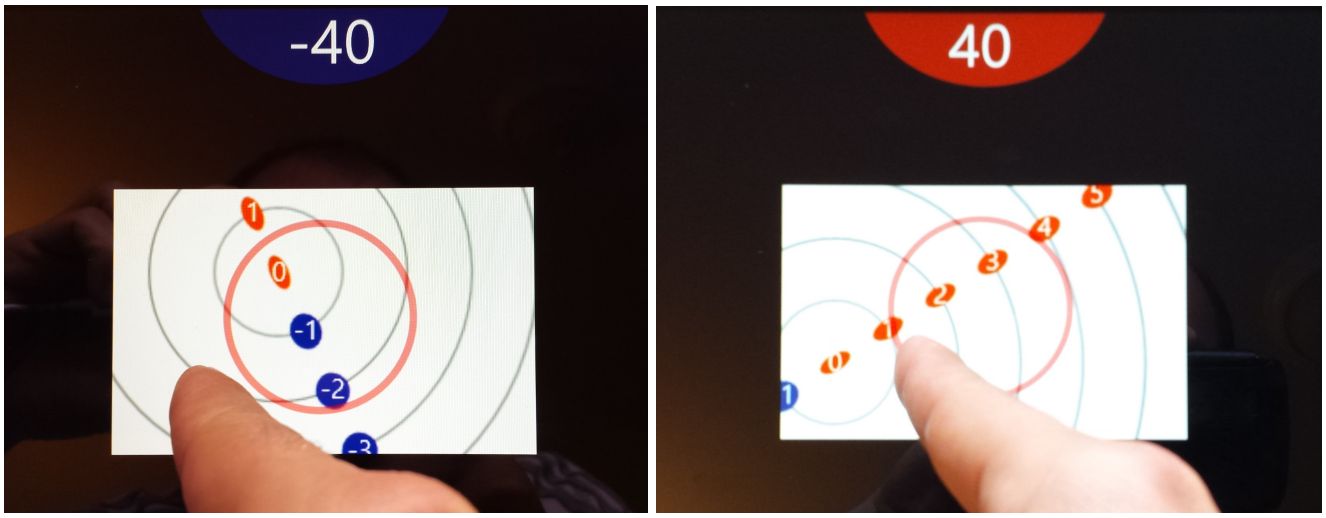


Figure 20: Two instances of the experimental trial variant used in experiment three, with a negative target value on the left and positive on the right. Color coding of the indicated target value mitigates sign confusion, an inherent weakness in this type of experimental design.

task. In other words, the relation between trials and target values is a bijection.

The set of target values was conceived on three assumptions. First, the range should cover the presumed physical limit (i.e. a line length of one meter). Second, the target values should be equally spaced to provide interpretable results. Third, the steps should be numerically simple, under the assumption that humans do not relate odd-sized numbers very well to each other.

To take these assumptions into account, the 10 symmetrical target values

$$\pm\{10, 20, 30, 40, 50\}$$

were chosen as the target value set for full and evenly spaced coverage of the input range, with the step size (10) presumed optimal in terms of numerical comprehension.

Both the interfaces and task exercises were executed in the order presented here. For each interface, all target values were systematically randomized, using a unique string identifier for each participant as seed (for reproducibility while uniquely randomizing for each participant).

All in all, a total of 5 participants $\times$ 4 interfaces $\times$ 2 tasks $\times$ 10 trials $=$ 400 trials were recorded.

7.2.4 Procedure

Participants were given an introduction and demonstration to the general procedure for completing trial instances of the two task variants for both negative and positive numbers. Next, a few practice sessions were completed so as to get comfortable with the situation. Each participant was placed in front of the on-screen input space in a position that allowed for a natural and relaxed use of the interacting arm. In addition, participants were informed that the off-screen interfaces varied slightly, but no other hints were given, in order to obtain results that accurately reflect how well a given interface performs on the premise of the *user's* intuition. That is, the aim of the procedure was to prevent data pollution by ensuring that the user did not try to conform to any predefined notions of off-screen space.

7.2.5 Participants

Two males and three females agreed to participate in the experiment. All were in the age span 20-40 (mean 29, standard dev. 9.3), all had normal vision and all chose to interact with the right hand.

Figure 21: Happy and healthy Silicon Valley techies in their prime, all presumed quick to engage with new technology.

7.2.6 Hypothesis

It was hypothesized that a simple extension of the plane would best align with the users' perception of off-screen space, since the display itself is planar. Hence, the plane-normal interface was expected to yield the shortest interaction times (H1).

Adding directional capabilities, i.e. allowing the user to navigate a planar extension by anchoring to the directional projection of the finger, was assumed difficult to manage for two reasons. For one, directional projection obviously has little effect in close proximity to the display, due to the projection distances being short. On the other hand, the velocity was expected to become too sensitive outside the display, since the convex curve formed by the motion of the arm will increase directional projection distances very rapidly (from zero and to infinity for when the arm is parallel with the plane). Hence, the plane-directional interface was expected to equal the plane-normal interface for low target values, but overshoot for the high targets due to uncontrollable velocities, possibly with additional clutches for regaining navigational control and sensitivity (H2).

Introducing a curve in the off-screen space was presumed to maximize continuous (single-touch) travel distances while keeping the sensitivity constant. However, the transition in moving from a visible plane to an invisible curvature was assumed disorienting and with the user likely to follow a plane rather than the depth of the sphere. This would presumably result in undershooting of degree increasing with target value, making the spherical interface slow and clutch-based for high target values (H3).

Lastly, all off-screen interfaces were expected to perform less optimally for the side of the non-dominating hand due to more constrained movement and obstruction of view, as has been shown to be the case in previous work[10](H4).

7.2.7 Results

To reduce noise, outliers were detected and removed for each type of interaction, defined as the unique combination of an interface, dimension and target value.

Trials were identified as outliers if their duration violated an upper fence, which we define as the full interaction time exceeding three interquartile ranges above the upper quartile (Q3) or under the lower quartile (Q1). In addition, any trial taking above a hard limit of 25 seconds were bluntly categorized as an outlier[15]. In numbers, three trials from three separate participants were removed (plane-normal, horizontal: 1, directional, horizontal: 1, spherical, horizontal: 1).

Baseline

As seen in figure 22, the duration for the baseline could appear to be approximately linear in the target value, or put differently, linear in the distance to cover. In numbers, we see durations of approximately four through seven seconds. There is a slight tendency for the curve to flatten out at target extremums. A possible explanation for this may be found on the right-hand side of the figure, where it is seen that inertia covers almost all the travel for high targets and is presumably of high velocity. One particular trait of the inertia appears to be an immediate and rapid increase for positive target values, which correspond to the side of the participant's interacting hand. This is sensible, since the only way to avoid obstructing the display for an east-to-west swipe motion is to interact quickly (by removing the hand and letting inertia do the traveling).

Overshoot statistics is also shown by the figure, defined as the accumulated travel distance taking place after the target has entered and prematurely exited the bulls eye, over the total travel distance. As is seen, overshooting could appear to decrease slightly with increasing target values. This is somewhat counter-intuitive, as one may expect greater inertia velocities at higher target values and therefore more overshooting. However, the noticeable spike for the horizontal dimension and target 20 could possibly indicate that users tend to apply inertia in a very rough and invariant manner regardless of the target, which would consistently result in more overshooting for smaller (nearby) targets.

The statistic on the number of touches (which are shown in absolute values rather than percentage) indicate that user's perform a touch for every 0.1 meter. Thus, targets highs such as 50 require approximately five touch operations, which implies significant redundancy in the physical input for the baseline.

Plane-normal

In figure 23, wee see that the duration for the plane-normal bears strong resemblance with the baseline, although slightly more fluctuating and with an overall pattern of longer durations. Interaction now takes upwards of 12 seconds in general. Overshooting appears more arbitrary and difficult to interpret. The number of touches and redundant input is significantly lower however and only rises above a single one (the initiating touch) for target values of 40 or greater.

Plane-directional

The plane-directional interface results in the data shown in figure 24. Again, durations are significantly more fluctuant and take upwards of 11 seconds, slightly worse than the baseline. What stands out for this interface though, is the close-to elimination of subsequent touch operations. Conversely, there is significant overshooting for all intermediary values. While speculative, the

[15]Such an instance corresponds to *at least* 5 seconds for every interval of target value (0.1 meter) - clearly an extreme duration and one that has significant impact on the interquartile range, due to the smaller sample size.

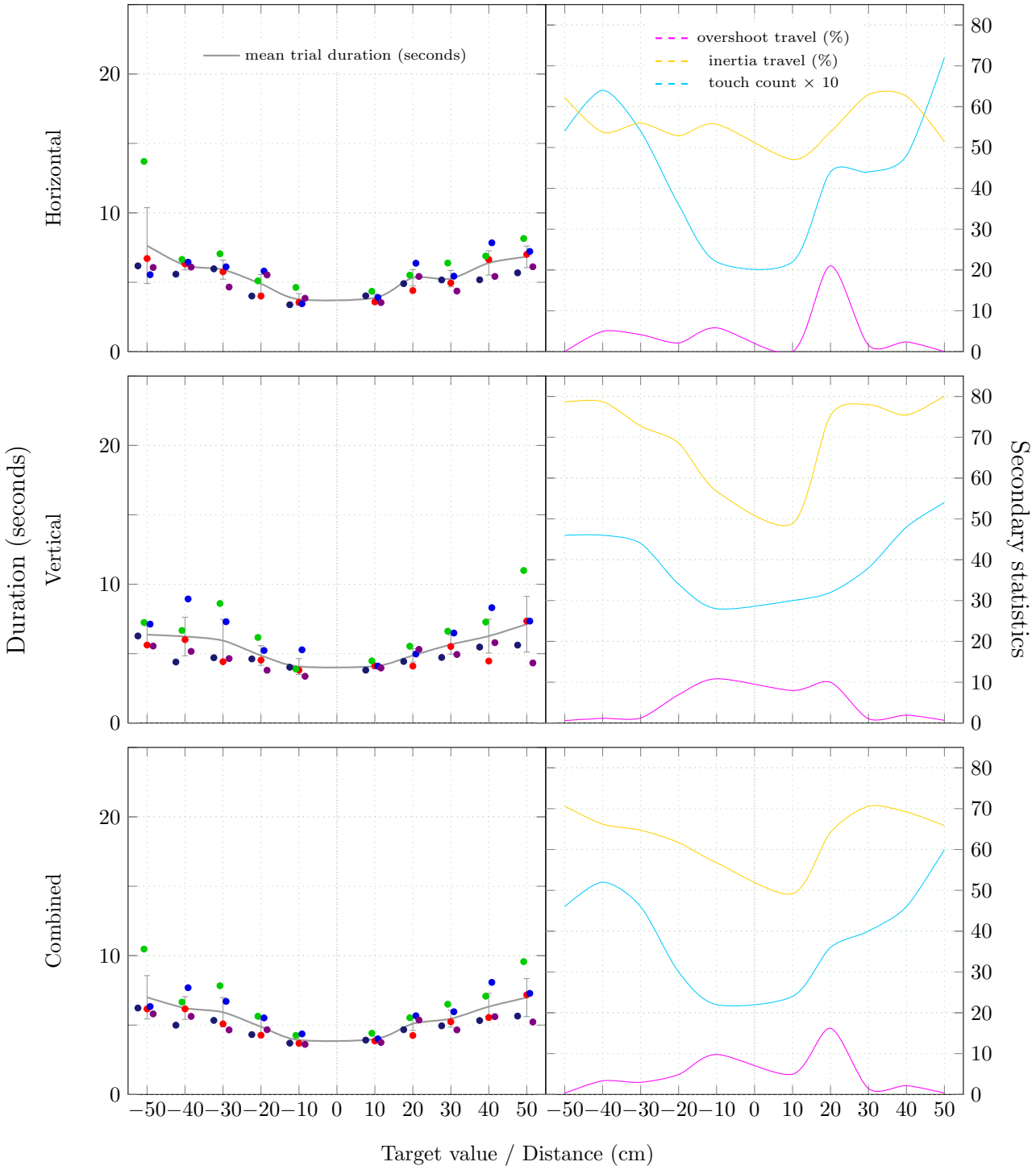


Figure 22: The baseline data collected for the horizontal dimension (top) and vertical (middle). The two combined is also shown (bottom), in order to obtain an idea of how a true two-dimensional interface is likely to perform. The left-hand side of each row shows the duration data in lightly scattered form, along with the mean (that does not include outliers). On the right-side of each row, statistics is shown for the mean percentages of overshoot and inertia travel, as well as the mean number of touches used (note the latter is in absolute values and is amplified by a factor of 10, to distinguish it more clearly on the scale used).

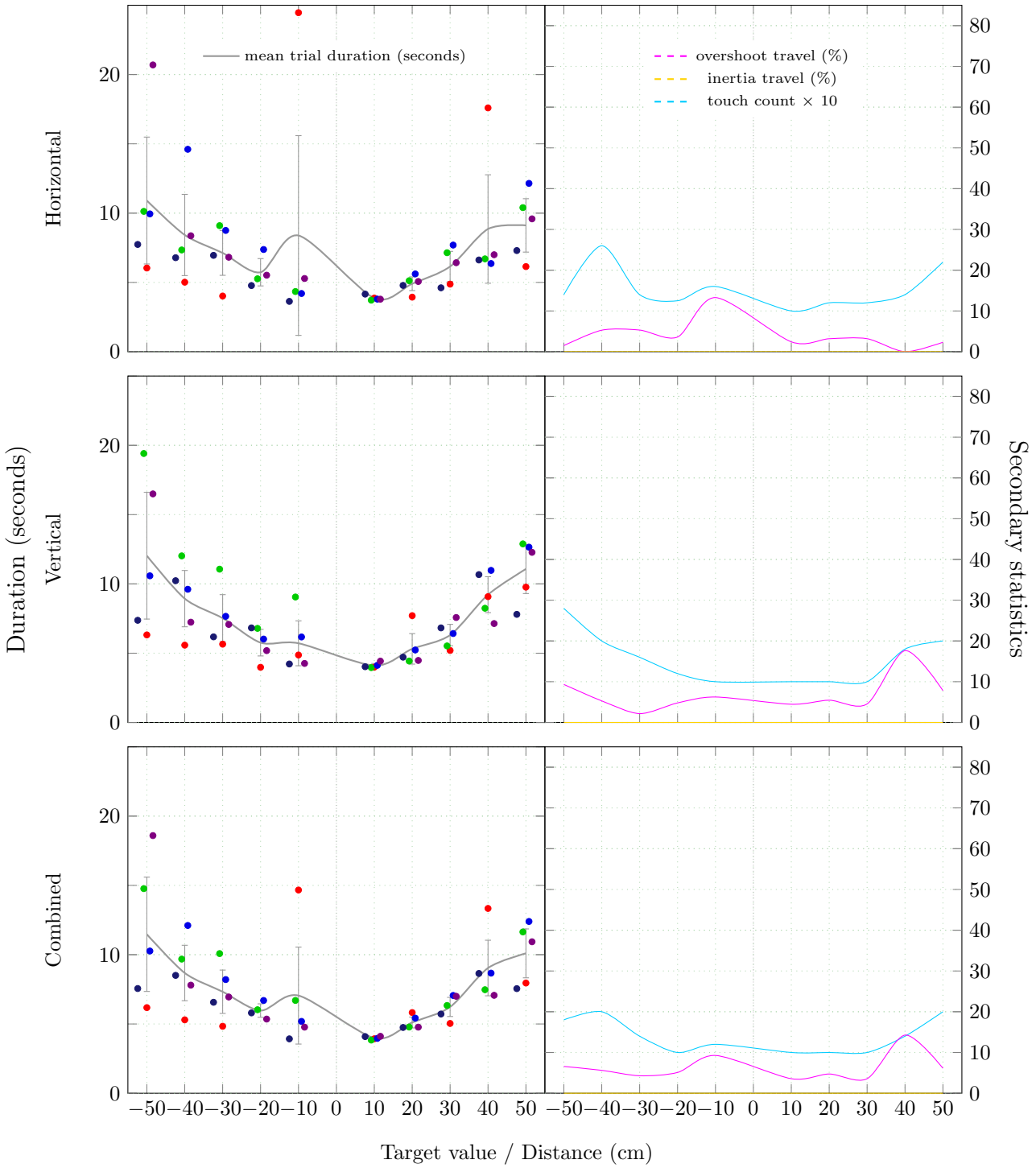


Figure 23: The data collected for the plane-normal interface. Inertia data do not exist here, as it is not a feature of any of the off-screen interfaces.

drop in overshooting at the extremes may possibly be a result of the user performing a release before reaching the target value hold. This would be supported by the concurrent rise in touch count for high target values and could indicate difficulty in controlling the interface at high targets. Not shown by the plots, but noted by two of the participants, is that the directional projections became more "jittery" at high target values, which again supports this theory.

Spherical

The data of the last remaining interface is illustrated in figure 25. This could appear to have the lowest variance of all three off-screen definitions. Also, the interaction durations are slightly better than the all past off-screen interfaces and come very close to the baseline, with an approximate maximum of nine seconds for extreme targets . Overshooting is almost non-existing and the same

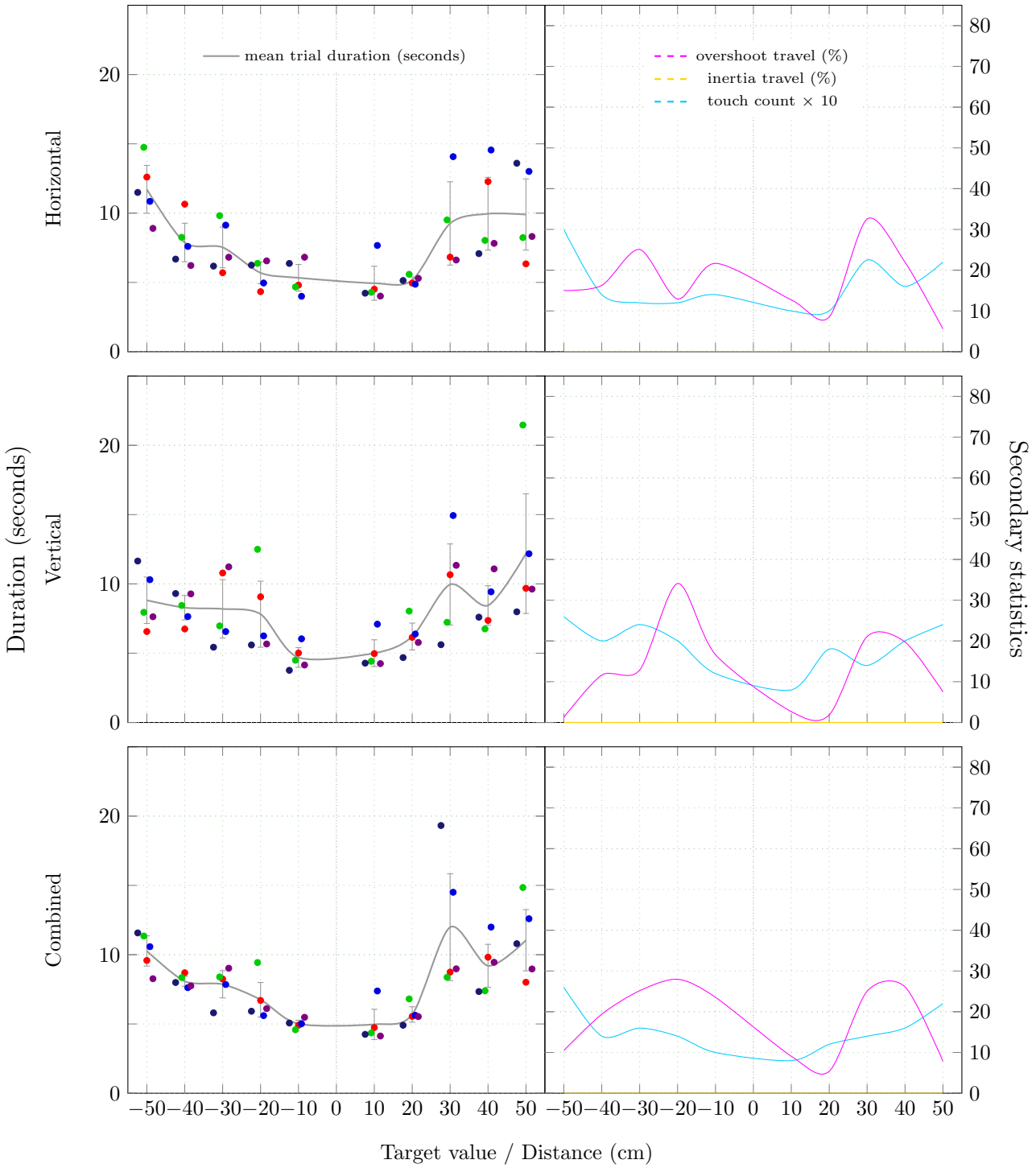


Figure 24: The data for the plane-directional interface.

applies to subsequent touch operations, there are almost none.

7.2.8 Discussion

To sum up and evaluate, we will compare the general behavior of each interface, as captured by their means. Furthermore, we will view the combined dimension as the most meaningful to compare, under the assumption that future use of off-screen space is likely to be for two-dimensional interfaces. Such comparison is given in figure 26. From this, it appears that the spherical interface aligns best with the baseline and is most stable. We therefore dismiss the original idea that the plane-normal should perform optimally in terms of duration time (H1).

For the plane-directional interface, this did indeed result in significant over-shooting, increased touch counts and was inferior from a time-perspective. This supports the initial presumptions about the interface (H2), which

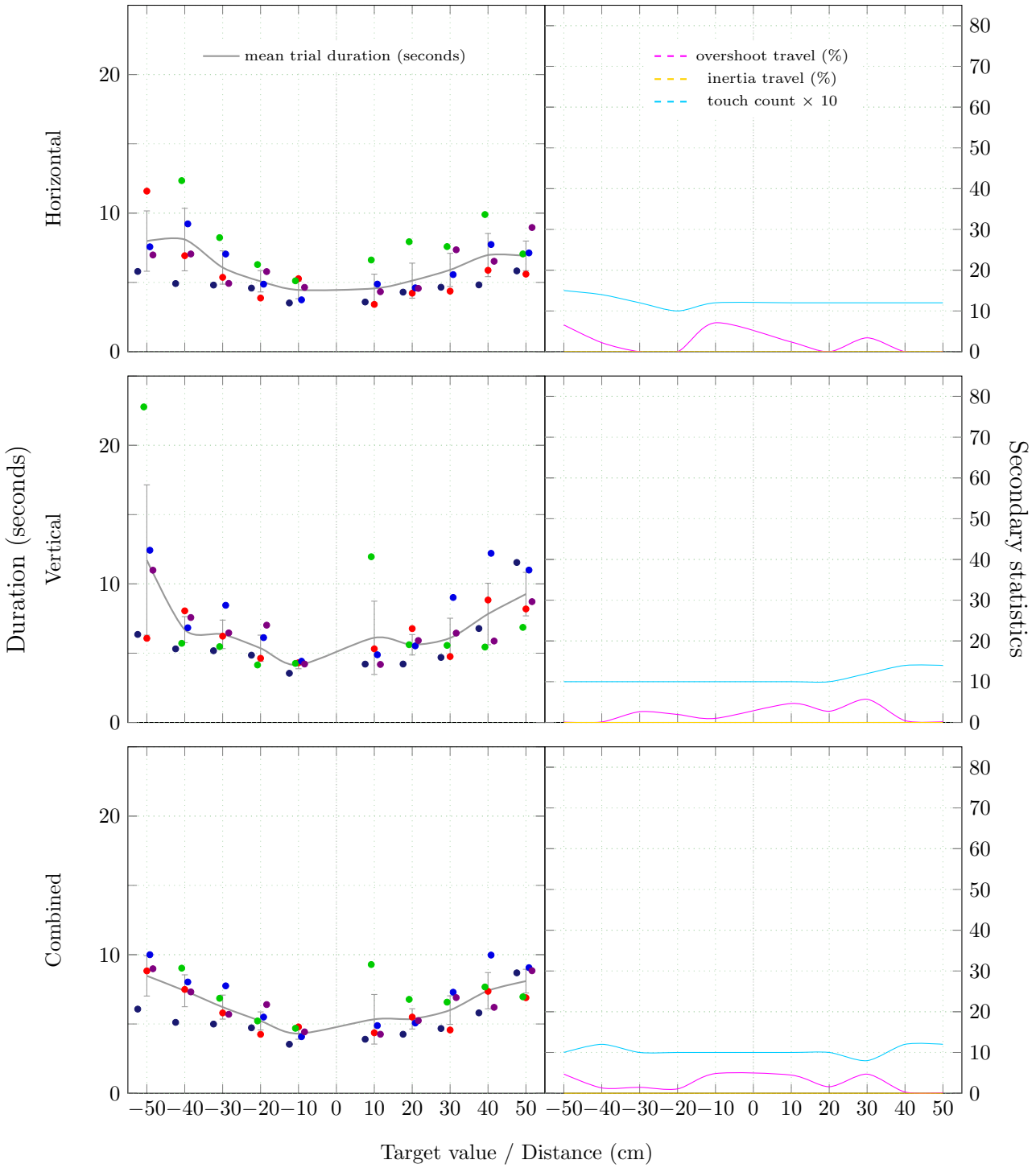


Figure 25: The data for the spherical interface.

are therefore confirmed.

Contrary to what was expected, the spherical interface did not suffer from long durations and came quite close to the baseline. In addition, this interface had virtually no touch input redundancy and almost zero overshooting. This hypothesis (H3) is therefore rejected.

As previously touched upon, a right-handed interaction may possibly obstruct part of the view for when the user initiates an on-screen swipe that ends in the north or west. With regards to the test interface, this corresponds to seeking out high target numbers in the horizontal dimension and low target numbers in the vertical. Thus, the data for these targets should show disproportionately longer durations, if the assumption of negative impact is true. However, the data does not indicate any significant pattern that strongly supports this view. This leaves the general assumption (H4) rejected for now and any minor effects to be explored more intricately in future work.

On a final note, it is assumed that the baseline has

been significantly advantaged in this experiment, since the participants are all experienced with that interface and conversely, are all new to the idea of using off-screen input. Hence, the baseline may have appeared best in terms of duration, but the picture could likely be different, had all interfaces been on par in terms of user experience. In addition, all off-screen interfaces are presumed to greatly enhance the user experience by eliminating redundancy in the input, as was observed here. This allowed participants to locate target values in a single motion (except for the extremums that occasionally required an additional touch operation). Whether any incurred strain of off-screen interaction have a negative impact, such as the physical fatigue identified in previous work[28], is yet another topic left for future research.

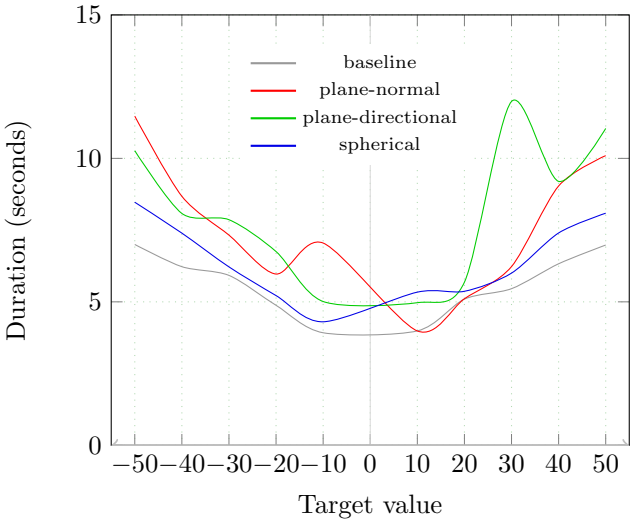


Figure 26: The mean for the baseline and all off-screen interface definitions. These are the combined means of the perpendicular horizontal and vertical dimensions, which are presumed to provide good insight into the interaction efficiencies of a true two-dimensional interface.

7.3 Experiment two: learning the sphere

Having determined the spherical interface as the most optimal, it was desired to explore this spatial definition in more detail. That is, we would like to learn how well users' expectations of the space relate to the idea of a simple sphere. Participants were therefore asked to complete a variant of the task that systematically investigates their perception of spatial targets versus their the true projections, as currently defined by the spherical interface.

7.3.1 Tasks

Participants were asked to perform estimations in a two-dimensional task, for combinations of target values and line angles that correspond to dividing the entire spherical input space into a circular grid. This grid and distribution of target values is shown in 27. The task was then to position the input (finger) on the spatial location that corresponds to the requested target value on some line angle, as the participant perceived it.

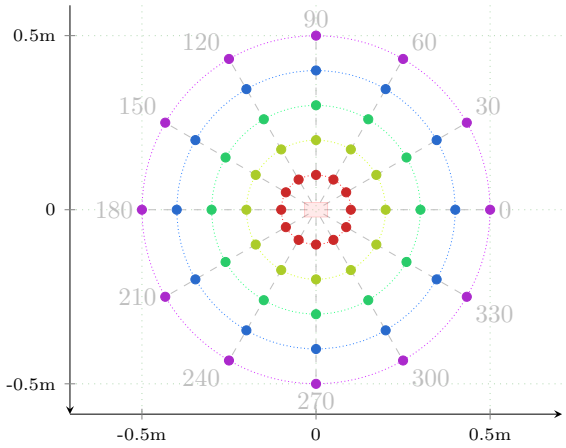


Figure 27: The mesh of target values for experiment two, defined by the dividing of the input space into a circular grid with origo at the zero target value. As can be seen, the grid has radials for every 30 degrees and target values for every 0.1 meter. The rectangle at the center corresponds to the exact dimension of the on-screen input space (as derived during the calibration process). Both major grid axes align with the device form factor, such that the viewpoint corresponds to a top-down view of the plane spanned by the device. Note the color coding convention introduced to clearly distinguish equivalent target values, a visual aid that will be used recurrently henceforth.

7.3.2 Input interfaces

The participants only interfaced through the spherical definition of space for estimating targets in off-screen space. However, no visual feedback of the input was given, as the very objective was to estimate locations without any attempts to conform to the interface.

7.3.3 Trial design

The variation of the experimental task was identical to that used in experiment one, except there were now two dimensions and $^{180^\circ}/_{30^\circ} = 6$ differently angled input lines (each line corresponding to two opposing and aligned radials). For the angling of any line, the positive range was placed on the top-most radial and the negative range on the opposing bottom, under the assumption that this ordering was intuitive to most users. Also, all combinations of angles and target values were randomized using the same seeding technique as in experiment one.

Although no visual feedback was to be given, some indicator of trial initiation was considered meaningful. To provide this in an intuitive manner, the movement of entities was reversed, such that the bulls eye now followed the user's input, with the trial line remaining fixed instead. Hence, to initiate an estimation trial, the participant touched the screen once, after which the bulls eye began to follow the finger, quickly sliding out of view as the user's finger moved outside the boundaries of the display (to perform the estimation).

A total of 5 participants $\times$ 1 interface $\times$ 6 angles $\times$ 10 targets were estimated, yielding a total recording of 300 trials.

7.3.4 Procedure and participants

Similarly to the previous experiment, the same set of participants were given demonstrations, practice trials and clearly advised that their intuition of target locations was the primary intention of the experiment, rather than completion time. Participants touched the on-screen space to initiate a trial, then moved to their chosen point of estimation and used a clicker to indicate completion with the non-interacting hand.

7.3.5 Hypothesis

It was hypothesized that users' assumption of space is closely tied to the spherical motion of the arm, but that actual interaction represents some departure from this ideal, due to the interplay of complex and unknown factors. That is, reality is presumed much more intricate for a variety of reasons, such as obstruction of view and the complex interactions between the joints of the arm.

7.3.6 Results

To obtain an initial grasp on the three-dimensional data and any potential patterns contained within it, we will seek to perform various visualizations.

View of raw data

Figure 29 shows the data color coded for each participant and from a multitude of angles. Here, it is seen that the input for all of them do in fact follow some degree of curvature around the device, both for the horizontal and vertical dimension. The data sets also differ significantly. Two of them, as indicated by violet and blue, provide vastly different takes on the input space. The first provides a close to planar input and the latter a more spherical. One of these even goes well beyond the extremums of the scale. This has a reasonable explanation however, as the participant was observed moving a few steps from the device, in order to provide these off-scale inputs. While being an instance of unintended user behavior, it was left uninterrupted - and none of the other participants took similar "off-road" approach.

Also seen from the top-view (bottom) of the figure, is that the input appear to be heavily skewed towards the south-west corner of the plane spanned by the device. Conversely the north-west corner is noticeably void of points. In addition, the side-view (center row) clearly reveals that participants are quick to depart out from the plane normal in the northern region, but much less so in the south. This could possibly be due to the obstruction of the body and a reluctance to pull the hand up towards the chest area, since that is an awkward and strenuous motion. These observations combined suggest a space that is not centered at the exact origo and one that does not have equivalent shapes in the northern and southern regions of the off-screen space.

We may also want to investigate the data color coded by target values instead, as shown by figure 30. From here, it is seen that the estimations vary little for small targets, but slowly increase and become severely scattered for the extremes. This trend of dispersion appear most pronounced in the south-east corner. Correlating this with the previous figure 29, it could appear that participants struggle more or less with achieving consistent estimations in this south-east region.

View of projected data

To expand on the analysis, we now relate the data to their resulting projections in the spherical interface, as shown in figure 28. If the spherical interface has been used in an ideal fashion, the resulting locations should show a higher degree of dispersion, compared to the "raw" top-down view, due to the added effect

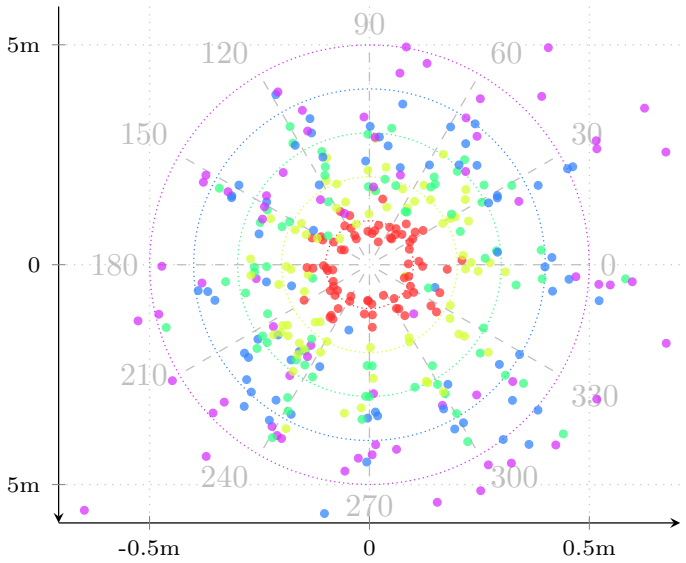


Figure 28: The estimated locations obtained from projecting each data point using the spherical interface, here color coded by absolute target value. As opposed to the top-down view of the "raw" data, height now has influence on the end location, which makes for a slightly different picture. In addition, the projection center is now the projected location of the shoulder onto the plane spanned by the device. As such, the device is not exactly in origo from this (projection) viewpoint, but should theoretically be slightly dislocated towards the north-west quadrant (since all participants are interacting with the right hand).

of height[16]. What we see instead though, is somewhat equivalent density on the south-west to north-east line and a pattern of slightly more compaction on the perpendicular north-west to south-east line. This may indicate that the true space is compressed. To expand lightly on this, at least one likely contributing factor would be easy to identify: the joint of the interacting arm does not have constant flex. More precisely, the arm will have some natural flex when pointing directly at the screen, but slowly extend and eventually reach zero as the arm travels from the origo and towards the extreme target values. This effectively corresponds to a sphere of greater radius than defined by the interface, causing the user to undershoot slightly (which translates to greater compaction in the projections, as visualized here).

[16]To ensure that the reader follows, the two-dimensional view of spherical projections has an analogue in the "stretching out" of a globe onto a plane, which results in amplification of surface area. Hence, any point not in the center of the sphere will obviously have height greater than zero from the device plane and therefore show greater dislocation from this center, when compared to a top-down view of the same data in the actual space.

Figure 29: The total set of collected estimations with respect to the axis formed by the device and with the data color coded for each participant. The on-screen space is shown as a red rectangle and with choice of axis as if the device is lying flat down in the origo. The z-axis then represents perpendicular distance from the device plane and the x and y axes represent horizontal and vertical dimensions respectively. To provide spatial orientation of the data patterns, three different views are shown. The top row present a perspective on the horizontal dimension (as if the viewer is kneeling down and viewing the device in profile). The middle row inspects the vertical dimension (as if viewing the device in profile from the right-hand side). The bottom row presents a top-down view (as if the view point is straight on the device).

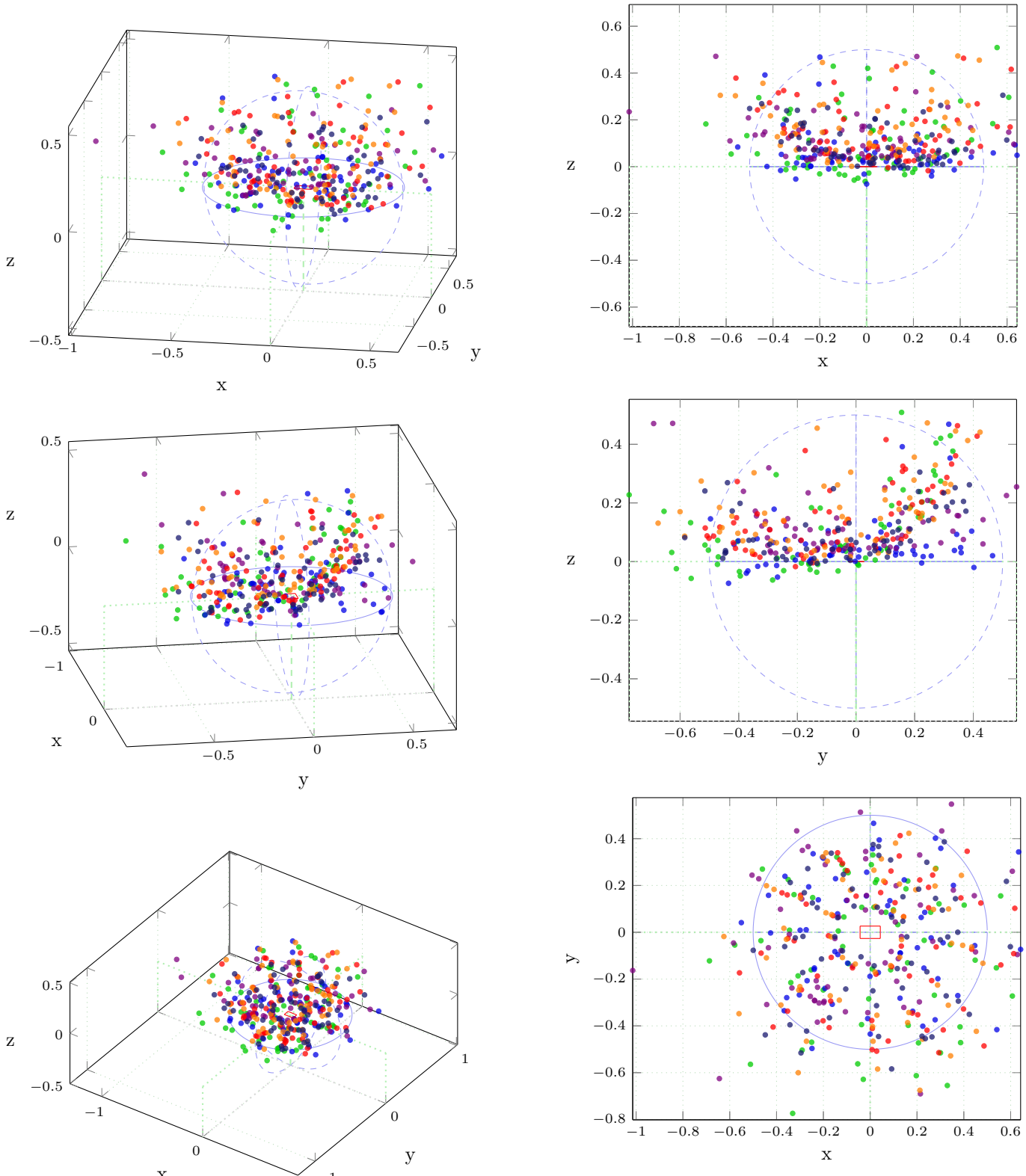
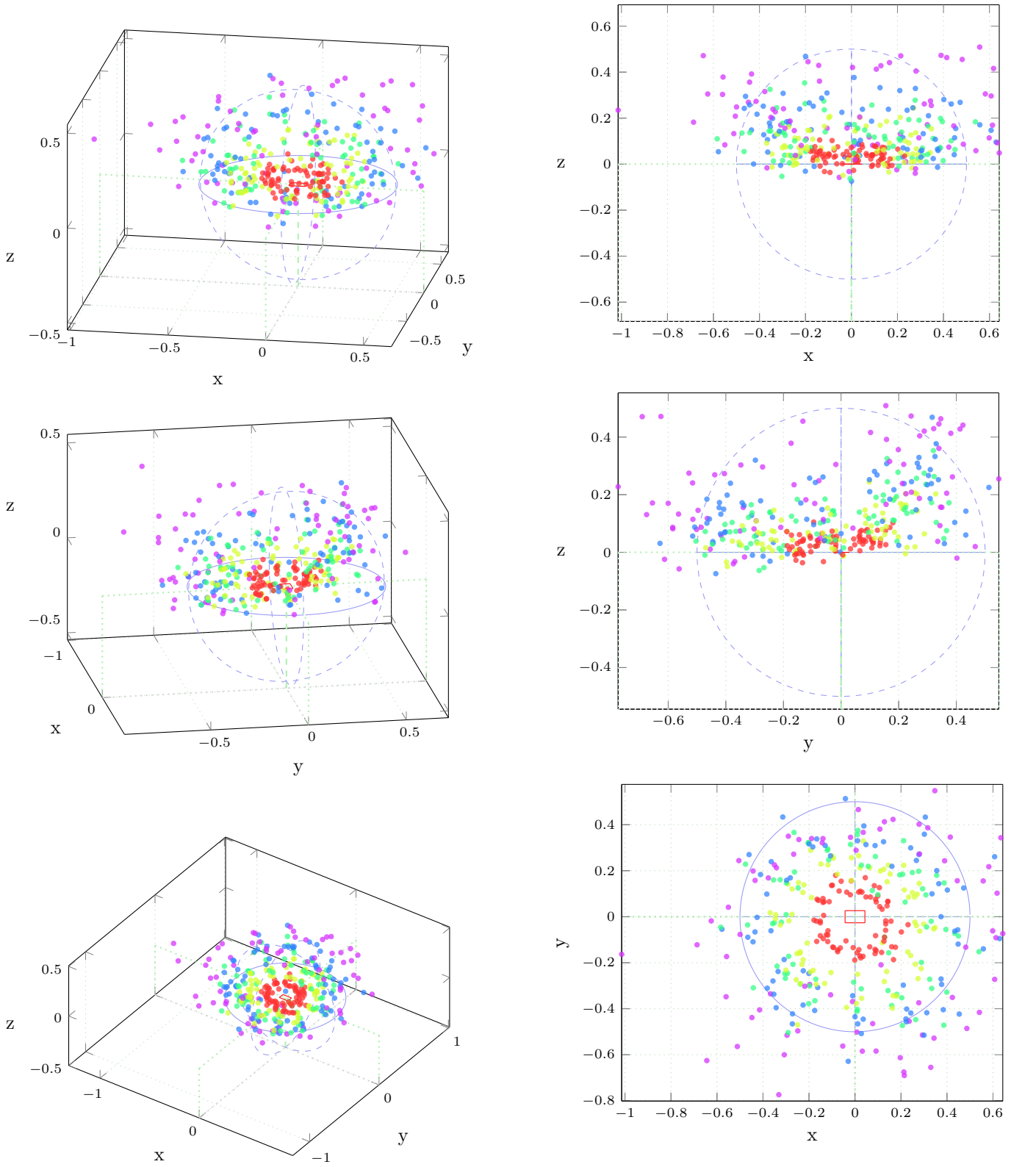


Figure 30: As the previous figure 29, but with color codings for each (absolute) target value.



Mean projection deviations

Attempting to generalize on the projection data by viewing all data points at once is overwhelming. Instead, we may derive and visualize the mean target deviations and how much "off" they are from the corresponding grid targets. This is illustrated in figure 31, which shows that there are indeed large differences. Notably, the participants appear to estimate too far right of the 90 degree radial and too far left on the opposite 270 degree radial. Such symmetry also applies somewhat to other angles as well and in a systematic way that suggests spatial compression and rotation. Also apparent, is the reluctance of participants to interact in the far north-west corner, as is implicit by the means of this region all being closely compacted and in close proximity to the origo.

To gain a better sense of the magnitude of the deviations, we will investigate using both intuition and hard numbers. Figure 32 provides the former in the form of a visual overview and table 1 the latter by the listing of numerical values. From the visual impression, the correlation between the estimation error and target values could appear linear, although clearly not applicable to all the data. In general however, the estimations begin to fluctuate with increasing proximity to the target range extremities, indicating that these may have been out of reach. Three of the participants gave explicit feedback in support of this view. Furthermore, the numerical data show almost a full doubling in mean deviation between the two extreme targets (-40 versus -50 and 40 versus 50). Also, the fluctuations could seem to occur for all but the compact north-east region and the trend becoming severely pronounced in opposing south-east region. Worst appears to be the latter, which shows a full 0.27 meter deviation from the extreme target (-50) on the 330 degree radial, supporting the idea of significant variance in that region. Hence, with the general pattern of overly pronounced fluctuations in the vicinity of extremums, these outer targets (40 and 50) are likely beyond the physical reach of the participants.

Target\Angle	0	30	60	90	120	150	Total
50	0.20	0.15	0.14	0.20	0.21	0.16	**1.06**
40	0.07	0.08	0.09	0.13	0.11	0.12	**0.60**
30	0.11	0.06	0.10	0.09	0.07	0.09	**0.52**
20	0.05	0.04	0.05	0.06	0.05	0.05	**0.30**
10	0.04	0.02	0.02	0.04	0.02	0.03	**0.17**
-10	0.02	0.02	0.03	0.02	0.03	0.05	**0.17**
-20	0.05	0.06	0.08	0.11	0.10	0.07	**0.47**
-30	0.07	0.08	0.06	0.07	0.06	0.13	**0.47**
-40	0.10	0.10	0.08	0.11	0.06	0.13	**0.58**
-50	0.18	0.18	0.17	0.15	0.11	0.27	**1.06**
Total	**0.89**	**0.79**	**0.82**	**0.98**	**0.82**	**1.10**	**5.40**

Table 1: The mean deviations and their various accumulations (the reader is reminded that positive targets reside in the upper/northern half of the space and negative those in the bottom/south by design).

Principal components for each target

As has now been determined, the participants' perception of space is indeed spherical, but does not follow the input interface in the ideal manner. In more detail, there could appear to be some compression, rotation and translation from the origo.

To approach a more concrete interpretation of the data, we will employ the heavier tool of principal component analysis. We will assume Gaussian noise for the fluctuations that occur around the absolute target values, i.e. the "rings" formed by the points for each such data set. In addition, we'll discard outliers based on deviation from the target, i.e. the distance from the estimation and to the target location. An outlier analysis was therefore performed for each grouping of trials (with equivalent angles and target values). Extreme cases were identified, understood as three interquartile ranges above the upper quartile (Q3) or below the lower (Q1). These outliers were not removed, but rather replaced with the mean of the non-outliers, so as to retain the weighting of the data points in the analysis[17].

Deriving the principal components for each target value results in the picture shown in figure 33. As can be seen by the means, there is a pattern of downward dislocation that increases by target value. This indicates that the location of the mean is in fact dynamic, in that it depends on the distance from origo. As for the observed scale and rotation, the components of the smaller targets do not exhibit any significant structure. The outer does, however, and paints a clear picture of a space that is rotated 34.6 degrees, which appear to align well with the visually observed patterns of the data.

7.3.7 Discussion

The data collected, and the interpretations that were derived from it, did indeed indicate a space much more complex than what is captured by the simple definition of a sphere, as was the hypothesis. Hence, the spherical interface may very well be a good approximation of the true input space, but also one that does not capture essential details, such as compression, rotation, scale and translation from the origo, all properties which the data suggest exist. Furthermore, the magnitude of these effects could appear to be correlated with the projection distance from the origo. That is, the space appears circular and centered at the origo for the initial small target values and expands into increasingly elliptical form with greater dislocation from the origo as the targets increase. This clearly makes conforming to such projection space less than straight-forward, because it then

[17] In numbers, four values were identified as outliers, corresponding to target/angle pairs $(-50, 150)$, $(-50, 120)$, $(30, 0)$ and $(40, 120)$.

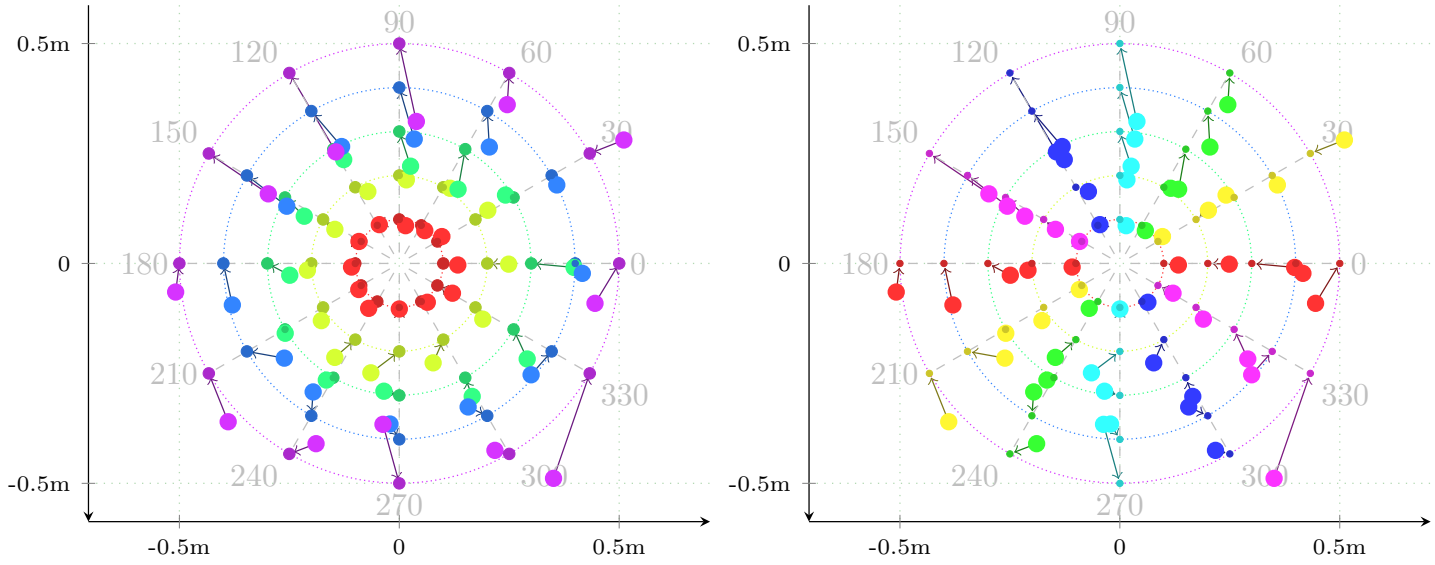


Figure 31: The mean estimations of target values versus their "true" locations in the grid. Each point represents the averaged estimation location for the particular angle and target value. On the left-hand side, each estimation mean is color-coded by target value and an arrow indicates the error, i.e. the magnitude and direction towards the true location. The plot on the right-hand side is identical, except color coding is by line angle instead.

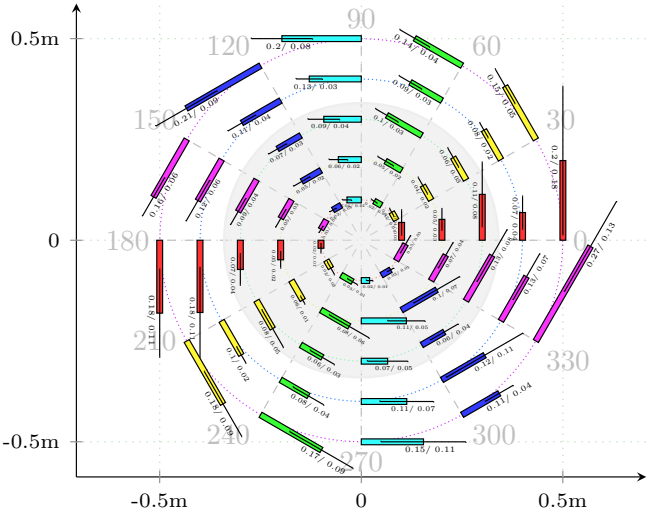


Figure 32: A compact illustration that captures the mean deviations and corresponding confidence values (with deviation defined as Euclidean distance from estimation to target). All bars are color coded by angle, with each capturing the mean of the sum of deviations for the target at its base. Hence, these magnitudes then also capture any scatter around the true location, as opposed to the previous figure 31. The numeric values for mean and confidence values are given underneath each bar and in that order.

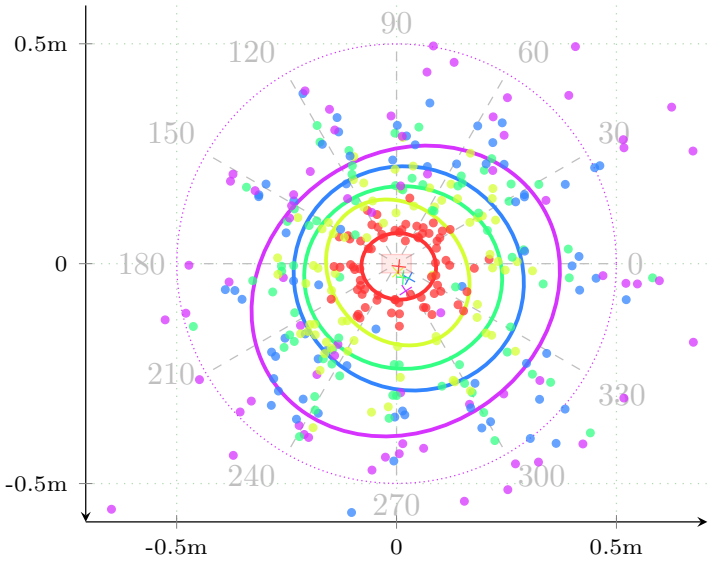


Figure 33: The result of applying principal component analysis to five sets of point projections, formed by grouping on absolute target values. The intersects of principle components for each set is shown by the perpendicular lines (crosses) at the corresponding means.

has a "third dimension" of dynamic curvature. Also, the input appeared to become fluctuant and incoherent for the extreme target values (40 and 50), possibly indicating that these are physically out of reach. As such, the experiment has confirmed the hypothesis, but leaves any more exact definition of the true input space open for interpretation.

7.4 Experiment three: optimized sphere

With experiment one having identified the spherical interface as the superior of the three examined and experiment two showing it as not capturing essential details, an attempt at evaluating some optimization would be desirable. The aim in this experiment was therefore to compare the spherical space against some modification derived from it. That is, the original baseline of touch-only is not of direct interest here (as its general performance versus a spherical off-screen interface has already been shown by experiment one). A modified spherical interface was therefore derived on the basis of the data from the previous experiment two and this was experimentally evaluated against the non-modified version.

7.4.1 Task and interface

Participants were requested to seek out target values as in experiment one, but now using a true two-dimensional task and a target grid similar to that of experiment two (as shown in the previous figure 20). As learned from experiment two, the extreme target value (±50) was likely beyond physical reach and therefore omitted. The second-largest target (±40) was then considered the new fringe value. Hence, the chosen grid pattern encompasses $^{180°}/_{30°} = 6$ lines, each with $^{40}/_{10} = 4$ targets.

7.4.2 Trial design

All target values were randomized in fashion identical to the previous experiments. A total of 3 participants × 2 interfaces × 6 angles × 8 targets were collected, through 288 trials in total.

7.4.3 Procedure and participants

Only three of the past participants could be made available for the experiment (mean 28.3, standard dev. 8.5), although the potential data was assumed adequate for making rough interpretations. Participants were informed that the space would vary slightly between the two interfaces, but were also requested to interact based on their intuition. No learning time was given, as the task and general interface was already familiar.

7.4.4 Hypothesis

It was assumed that the data from experiment two could be used to reach an optimization that would perform better in terms of interaction times, for all target values within physical reach. Hence, an elaborate interpretation of the past experimental data was undertaken to deduce the various properties that should be part of the optimized spherical interface.

Choosing general properties

The previous experiment had revealed effects such as compression, rotation and scale through the principal components of the aggregated data for each absolute target value. The strategy chosen here was then to modify the space by trying to conform to these, as opposed to elaborate attempts at theorizing on the complex interplay of factors that produce them (e.g. joint interactions, obstruction of view for certain inputs etc.).

To take these properties into account, it was first noted that when participants had been asked to estimate the extreme target (±50), they generally gave inputs that appeared to be at the maximum of their physical ability. Hence, the assumption was made that these estimations had in fact defined the natural boundary of the true space. That is, while being sincere attempts at providing estimations, they were just as much representative for what the participants perceived as the physical limit of extreme input. The data points for the extreme target were therefore presumed to be an indicator of a *comfort zone*, for which the participants felt they were able to interact within - and the corresponding principals components considered to be prime candidates for the modification.

Under the assumption that any curvature in the true input space is convex, we could choose to define the boundary of the modified space as the convex envelope of the data points of the comfort zone. Furthermore, it would be reasonable to assume that such boundary should be smooth, as opposed to jagged by the conforming to individually scattered data points. Hence, we may initially choose the ellipse formed by the principal components and eigenvalues of the comfort zone data as representative of the general shape of the true space. In visual terms, these components correspond to the outer ellipse in the previous figure 33, squeezed or stretched, as if it was a ball. Hence, this space becomes dispersed on the main principal component, compressed on the secondary component and skewed for all space not aligned exactly on either of these. In addition, the elliptical space spanned by these components is clearly significantly smaller than the original space and corresponds to an overall compression, if the boundary was still to represent the same extreme target value (±50). If not dealt with, this would change the spatial unit of scale.

Maintaining unit scale

To counter the latter, the assumption was made that the overall unit of scale should not change, but remain

fairly equal to that of the on-screen space. That is, the user should not experience any significant change in the magnitude of incremental movements when swiping into off-screen space. To align this with the choice of the comfort zone as the total input space, a new target value for its boundary was derived, such that the units of scale remained invariant. In other words, the comfort zone boundary should not equal the extreme of target values, but rather some value that provides a sense of an even and continuous space.

To derive this value, a simple approach was taken based on area. In the visual sense, if we "squeeze" the elliptical shape of the comfort zone back into circular shape, the target value of the boundary will be given by the radius of this circle. As shown by figure 34, this corresponds to a target value of 0.34 meters. This in return allows us to deduce new boundaries for all target values by simple scaling (i.e. their radii in the modified space). Hence, the optimized space that leaves the unit scale invariant is as shown in figure 35.

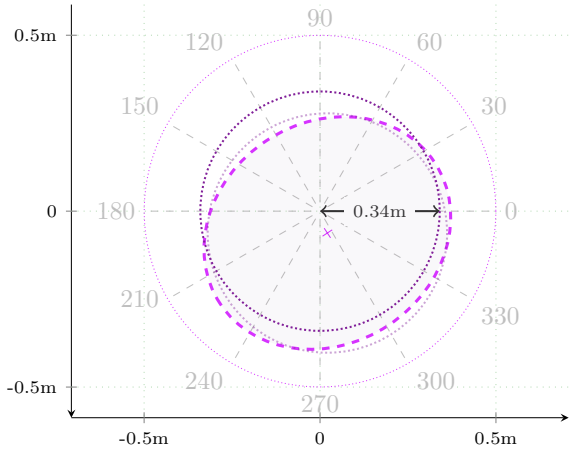


Figure 34: The derivation of the target value for the comfort zone boundary. Repositioning a circle (of area equal to the elliptical comfort zone) to the origo reveals a target value of 0.34. Adopting that value as the comfort zone boundary prevents odd scaling of the new input space.

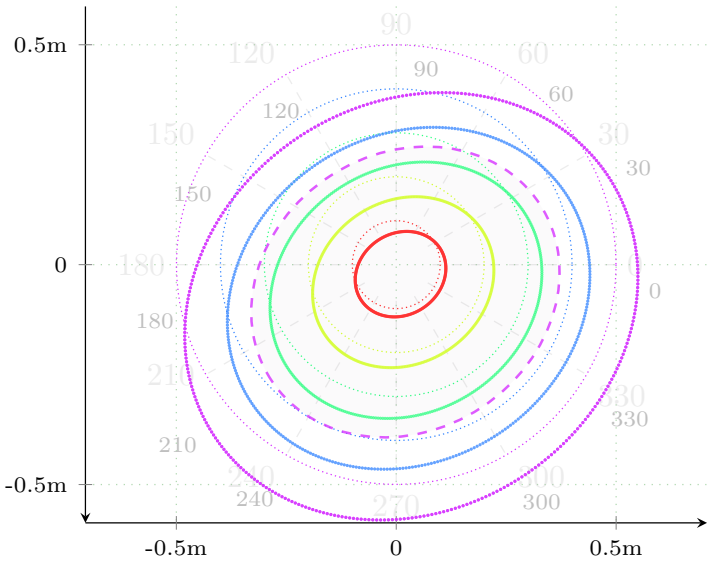


Figure 35: The scaled targets that result from adopting the newly derived value of the comfort zone boundary. The area enclosed by each new (elliptical) target boundary then has area equal to its old (circular) counterpart.

Defining a dynamic mean

Finally, we will complete the optimization by addressing the difficulty of a mean that converges in origo as the target value goes to zero (as remarked in the discussion section of the previous experiment). Visually, we would like to apply a dynamic "force", such that spatial locations are pulled towards the origo as they approach the mean. This idea is captured by figure 36. The result of such an operation is shown by figure 37, in which the imposed curvature leaves any projection invariant on what corresponds the the "old" extreme boundary (± 50), but have all converge to origo for when they approach the zero target value. Perhaps not surprisingly, the resulting space begins to resemble a three-dimensional view of a sphere, with one pole slightly shifted to the northwest, relative to the viewpoint.

Confirmation by visual inspection

Since we chose to derive the optimized interface from the target estimations, it would be natural to briefly investigate how the two compare now. Overlaying the past estimation data onto the newly modified space is shown in figure 38. While not a perfect match with the data, it could look as if the modification does a slightly better job at capturing the inherent patterns.

To abstract from the directions of the deviations, figure 39 compares the mean deviation magnitudes for all angles and targets. From this, it seems as if the greatest optimization is achieved on the 150/330 and 0/180 degree radials. This is numerically confirmed in table 2, from which these radials appear to have their mean deviations reduced with 17.02% and 18.67%, respectively. Also, in terms of target values, the initial target in the southern region (-10) shows a full improvement of 21.43%. A few deteriorations are also seen for some targets (20 and -40) but of much smaller magnitude, compared to the gains.

This completes the steps taken for reaching the optimization. The hypothesis is that this interface will better incorporate the observed complexities of the true space and as such, perform better than the original, from which it was derived.

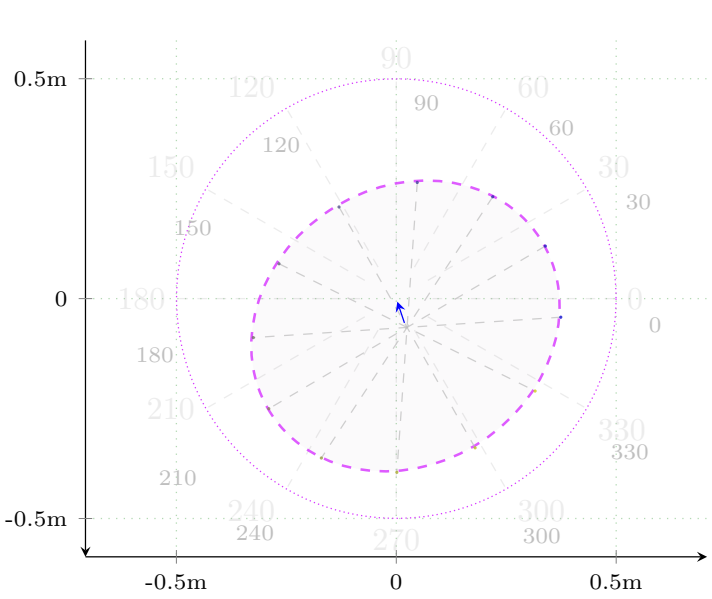


Figure 36: An intuitive illustration of the approach for dealing with the increasing dislocation of the mean, here shown for equi-distanced projections that correspond exactly to the comfort zone boundary. The blue arrow indicates the desired direction of this pull, which has maximum magnitude at the mean and should decrease linearly to zero at some target value of choice.

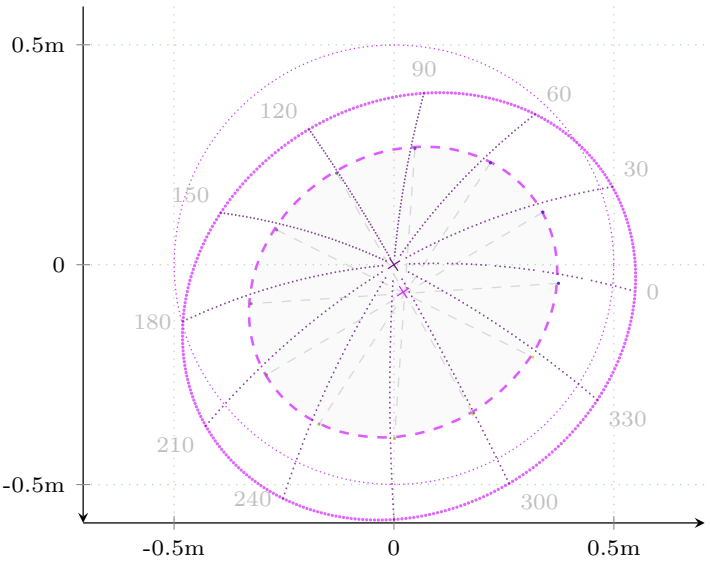


Figure 37: A visualization of the dynamic pull, that ensures that all projection space converges at the origo as target values approach zero, but yet leaves projections at an outer extreme boundary invariant. Projections inside this zone are then slightly affected, with the difference exemplified here by the "old" diagonals of the comfort zone (dashed lines in light gray versus the "new" curved radials).

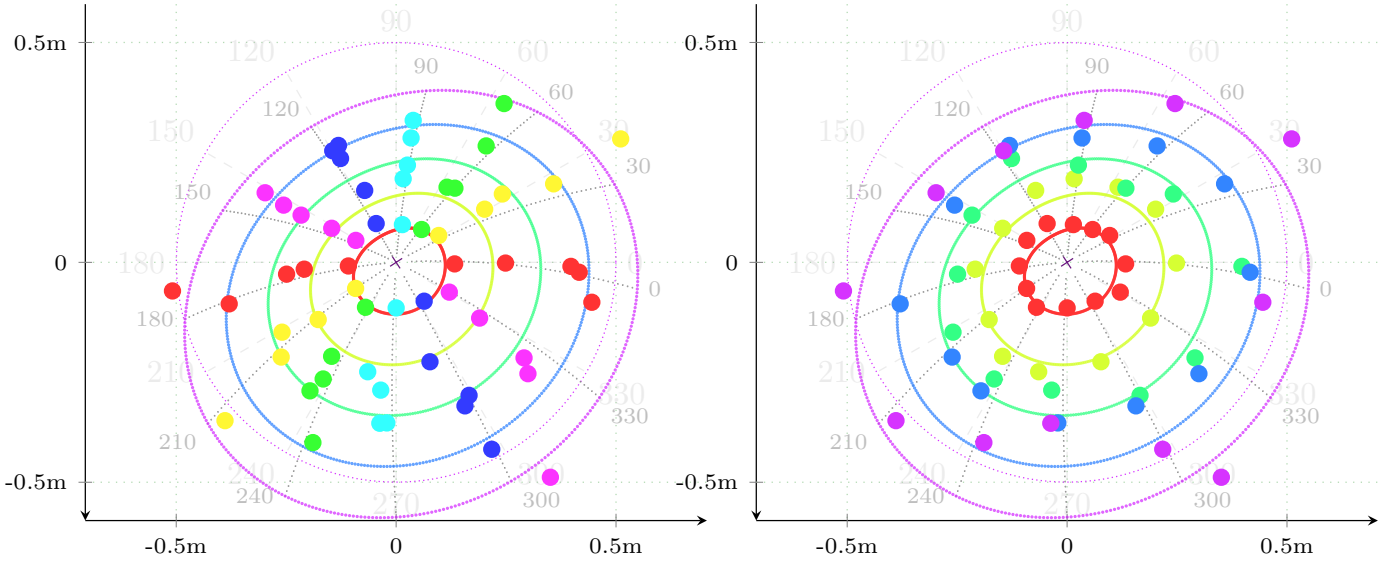


Figure 38: How the final optimization relates to the estimation means of the past experiment two, from which is was conceived. Again, each point represents the averaged estimation location for some particular angle and target value. Our primary interest is the data contained within the comfort zone (that is, all but the outer set of points). For these data, the modified space could appear to fare somewhat better.

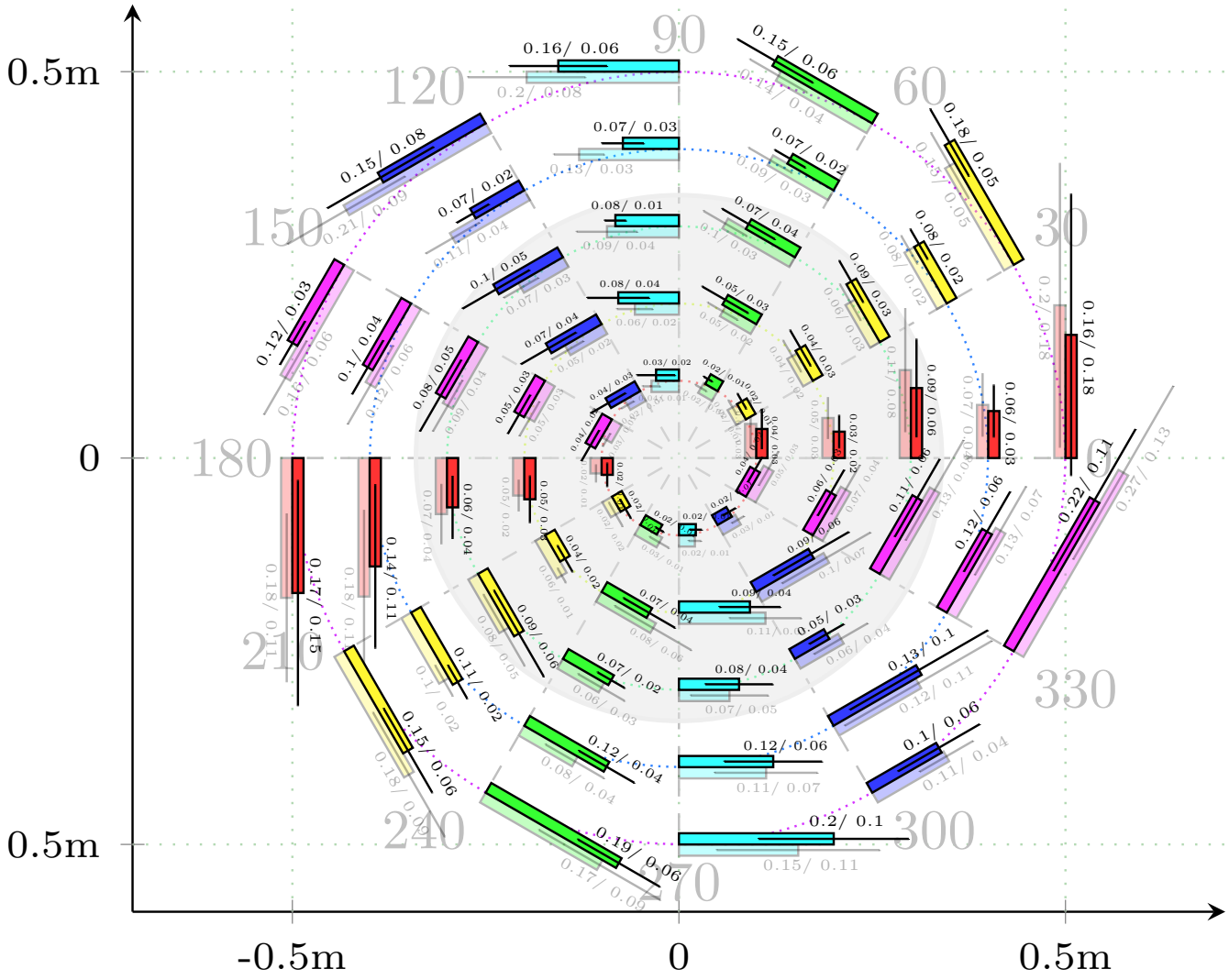


Figure 39: A revisit to the mean deviation wheel previously given in experiment two, but now showing the deviation means that would result if using the modified space. Again, each bar represents the mean of the sum of deviations for a particular angle and target value. Not all radials show improvement over the non-modified space (bars in fade), but the general picture does appear to be some reduction in deviation magnitude.

Target\Angle	0	30	60	90	120	150	Total
50	0.20	0.15	0.14	0.20	0.21	0.16	**1.06**
40	0.07	0.08	0.09	0.13	0.11	0.12	**0.60**
30	0.11	0.06	0.10	0.09	0.07	0.09	**0.52**
20	0.05	0.04	0.05	0.06	0.05	0.05	**0.30**
10	0.04	0.02	0.02	0.04	0.02	0.03	**0.17**
-10	0.02	0.02	0.03	0.02	0.03	0.05	**0.17**
-20	0.05	0.06	0.08	0.11	0.10	0.07	**0.47**
-30	0.07	0.08	0.06	0.07	0.06	0.13	**0.47**
-40	0.10	0.10	0.08	0.11	0.06	0.13	**0.58**
-50	0.18	0.18	0.17	0.15	0.11	0.27	**1.06**
Total	**0.89**	**0.79**	**0.82**	**0.98**	**0.82**	**1.10**	**5.40**

Target\Angle	0	30	60	90	120	150	Total
50	0.16	0.18	0.15	0.16	0.15	0.12	**0.92** +15.21%
40	0.06	0.08	0.07	0.07	0.07	0.10	**0.45** +33.34%
30	0.09	0.09	0.07	0.08	0.10	0.08	**0.51** +1.97%
20	0.03	0.04	0.05	0.08	0.07	0.05	**0.32** −6.25%
10	0.02	0.02	0.02	0.03	0.04	0.04	**0.17** +0%
-10	0.02	0.02	0.02	0.02	0.02	0.04	**0.14** +21.43%
-20	0.05	0.04	0.07	0.09	0.09	0.06	**0.40** +17.5%
-30	0.06	0.09	0.07	0.08	0.05	0.11	**0.46** +2.17%
-40	0.09	0.11	0.12	0.12	0.07	0.12	**0.63** −7.93%
-50	0.17	0.15	0.19	0.20	0.10	0.22	**1.03** +2.9%
Total	**0.75**	**0.82**	**0.83**	**0.93**	**0.76**	**0.94**	**5.03**
	+18.67%	−3.66%	−1.2%	+5.37%	+7.89%	+17.02%	+7.36%

Table 2: Then mean target deviations and their accumulations, for both the optimized and non-optimized space. The left shows the non-optimized estimation data, repeated from experiment two for comparison. The right shows how these deviations would look if the optimized space was to be used instead, with improvements indicated in percentage. Some radials and targets show great improvement, while only a few result in minor deterioration (again, the reader is reminded that positive targets reside in the upper/northern half of the space and negative are those in the bottom/south).

7.4.5 Results

Six outliers were removed prior to the data review, using a hard limit of no more than 25 seconds for the duration of each trial[18]. Data for both primary and secondary indicators of the user experience was collected, as enumerated in the following.

Primary indicators

In figure 40, an overview is given that concurrently relates the change in durations for all angles and targets. The initial impression from this appears as positive, but minor improvements on all angles, except for the 0/180 degree line. Two general and opposing contributions are easily identified for the optimized space: these durations start out with minor fluctuations, but seem to do modestly better than the non-modified space as targets goes towards the extreme (± 40).

In more exact terms, table 3 numerically reveals where the greatest change occur. This confirms that the fluctuations for lower targets ($\pm 10, 20$) do in fact underperform, but also that the modified space quickly picks up, with greatest gain in the south/lower half of the space (i.e. negative targets). There is significant gain on the lower extreme target (-40), possibly indicating that the modification has freed participants from colliding these physical input with their own body/waistline.

In terms of angles, the largest improvement (10.27%) is seen the 90/270 degree line and visually correlating this with the previous figure 40 indicates that the lower half of space is the sole contributor in this regard. Interestingly, the horizontal 0/180 line shows a slight deterioration (-6.05%) - although one could expect the opposite, given that the modified space appeared to visually align very well with the estimations from experiment two (as previously shown in the left-hand side of figure 38). Finally, the table shows an overall improvement of 5.24% reduction in interaction time, a modest number presumed highly affected by the fluctuations for lower-valued targets.

Secondary indicators

While interaction time has here been chosen to serve as the primary litmus test for the quality of the user experience, we will also give some attention to secondary indicators, as was done in experiment one. First, there are virtually no subsequent touch counts, as was also the case for the spherical interface in experiment one. However, there appears to be significant overshooting for the modified interface, as is depicted in figure 41 and numerically supported by table 4. In numbers, the modification does substantially worse, with a full 44.86% overall increase in overshoot travel. Furthermore, there is significant variance between the different angles and targets, making it difficult to identify any clear patterns. This variance is likely exacerbated by the low number of participants and outliers removal, making any elaborate attempts at interpretation somewhat speculative.

[18] In detail, these outliers correspond to target/angle pairs $(-20, 0)$, $(-10, 30)$, $(10, 60)$, $(10, 90)$, $(30, 30)$ and (40,0)

7.4.6 Discussion

With the new interface both visually and numerically appearing to slightly reduce interaction times, the proposed modification does enhance the user experience, although not of revolutionary magnitude. However, the achieved gains also appear to be accompanied by significant overshooting, although this is likely a result of the ordering of interfaces in the experiment: the modification was evaluated after the non-modified space, i.e. the participants were required to immediately adjust to the change of space. Since such a switch must necessarily be based on adjusting to visual feedback (given inputs versus observed incremental map movements), it would likely have some initial impact on navigational precision - which would be reflected by some overshooting. If this viewpoint is true, this overshooting would be most pronounced in the initial stages of navigating towards a target value, i.e. before the user gets the chance to adjust the input to the observed effects. This corresponds to small target values (such as $\pm 10, \pm 20$), which are exactly where the modified interface shows increased overshooting.

With interaction times still showing as superior in spite of (presumably experimentally incurred) loss of precision as reflected by overshooting, even faster interactions may likely result if the modified space was to be evaluated more thoroughly and in isolation. The modification is therefore assumed to be a small, but true step towards an optimization, thereby confirming the hypothesis. A more elaborate exploration of this approach for modifying a spherical space, preferably based on evaluating data from a much greater number of test subjects, is left for future work.

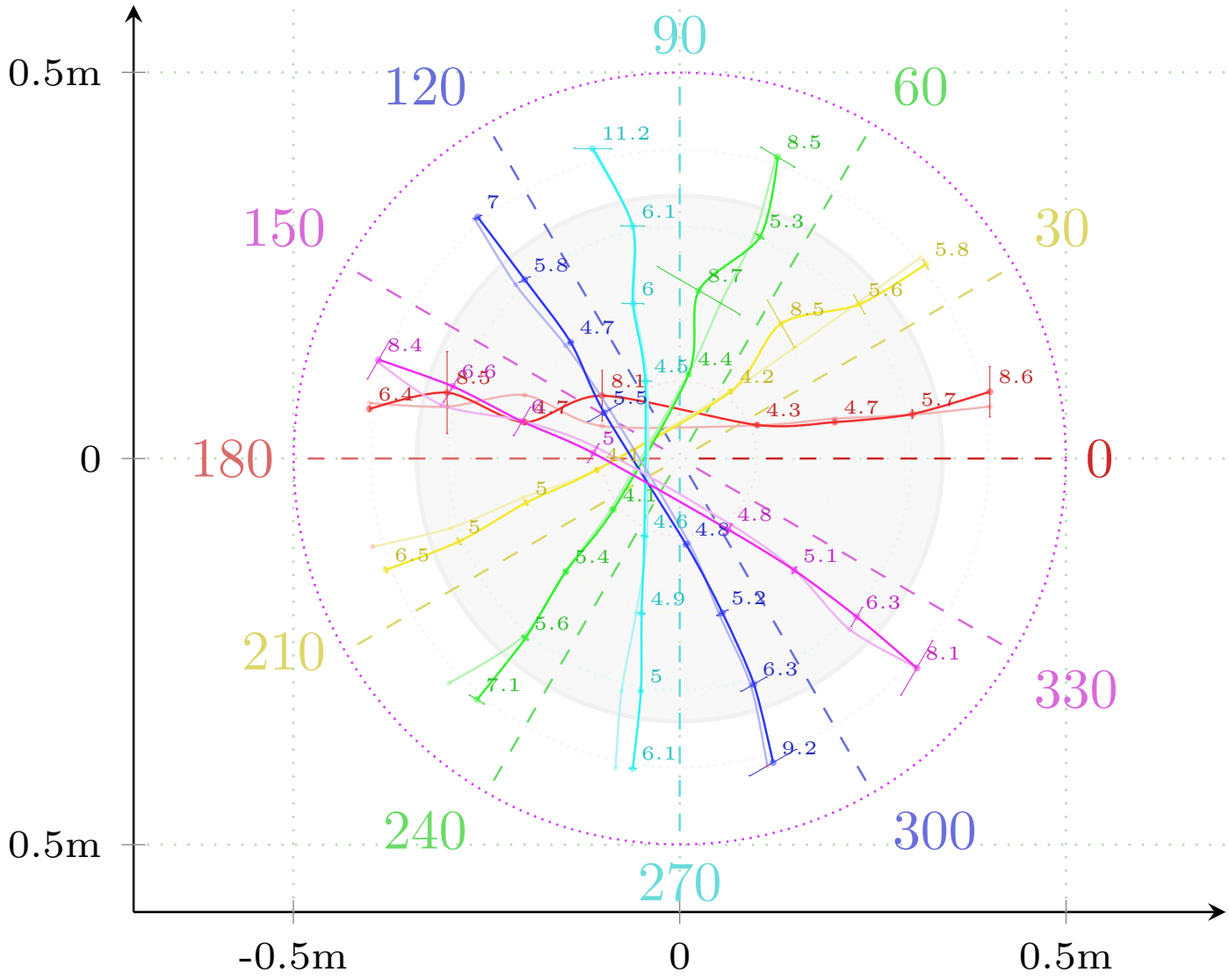


Figure 40: Differences in interaction durations between the optimized and non-optimized interface (the latter shown as faded). The duration for any angle and target is found by the perpendicular distance to the radial of same color. Duration scale is given by the numeric labels, shown for each target value of the optimized interface (in units of seconds). The two interfaces obviously come very close to each other, although with the optimized appearing to do slightly better overall as targets approach the new extreme (±40).

Target\Angle	0	30	60	90	120	150	Total
40	6.71	6.96	8.73	11.48	7.44	8.67	**49.99**
30	5.65	5.91	6.10	6.07	7.11	9.38	**40.22**
20	5.15	4.98	5.44	5.82	5.46	5.96	**32.81**
10	4.28	4.29	4.49	4.22	4.69	4.13	**26.10**
-10	4.19	4.16	4.43	4.25	4.41	4.08	**25.52**
-20	8.18	5.73	5.54	5.63	5.64	5.08	**35.80**
-30	6.69	6.98	6.02	7.50	6.71	8.15	**42.05**
-40	7.16	9.97	11.29	8.28	10.04	8.05	**54.79**
Total	**48.01**	**48.98**	**52.04**	**53.25**	**51.5**	**53.5**	**307.28**

Target\Angle	0	30	60	90	120	150	Total
40	8.62	5.85	8.48	11.20	6.98	8.38	**49.51** +0.97%
30	5.74	5.64	5.34	6.06	5.77	6.57	**35.12** +14.52%
20	4.69	8.52	8.70	6.04	4.69	6.02	**38.66** −15.13%
10	4.33	4.19	4.45	4.46	5.48	4.97	**27.88** −6.38%
-10	8.11	4.08	4.11	4.56	4.83	4.82	**30.51** −16.36%
-20	4.70	4.96	5.42	4.92	5.18	5.09	**30.27** +18.27%
-30	8.51	4.95	5.65	5.00	6.27	6.28	**36.66** +14.7%
-40	6.40	6.51	7.09	6.05	9.19	8.12	**43.36** +26.36%
Total	**51.10**	**44.7**	**49.24**	**48.29**	**48.39**	**50.25**	**291.97**
	−6.05%	+9.57%	+5.69%	+10.27%	+6.43%	+6.47%	+5.24%

Table 3: Mean duration times (seconds) between the non-optimized interface, left, and the optimization, right. As can be seen by the relative changes (as indicated in percentage on the right), there are minor improvements for almost all angles and targets, as is also reflected by an overall reduction in interaction time.

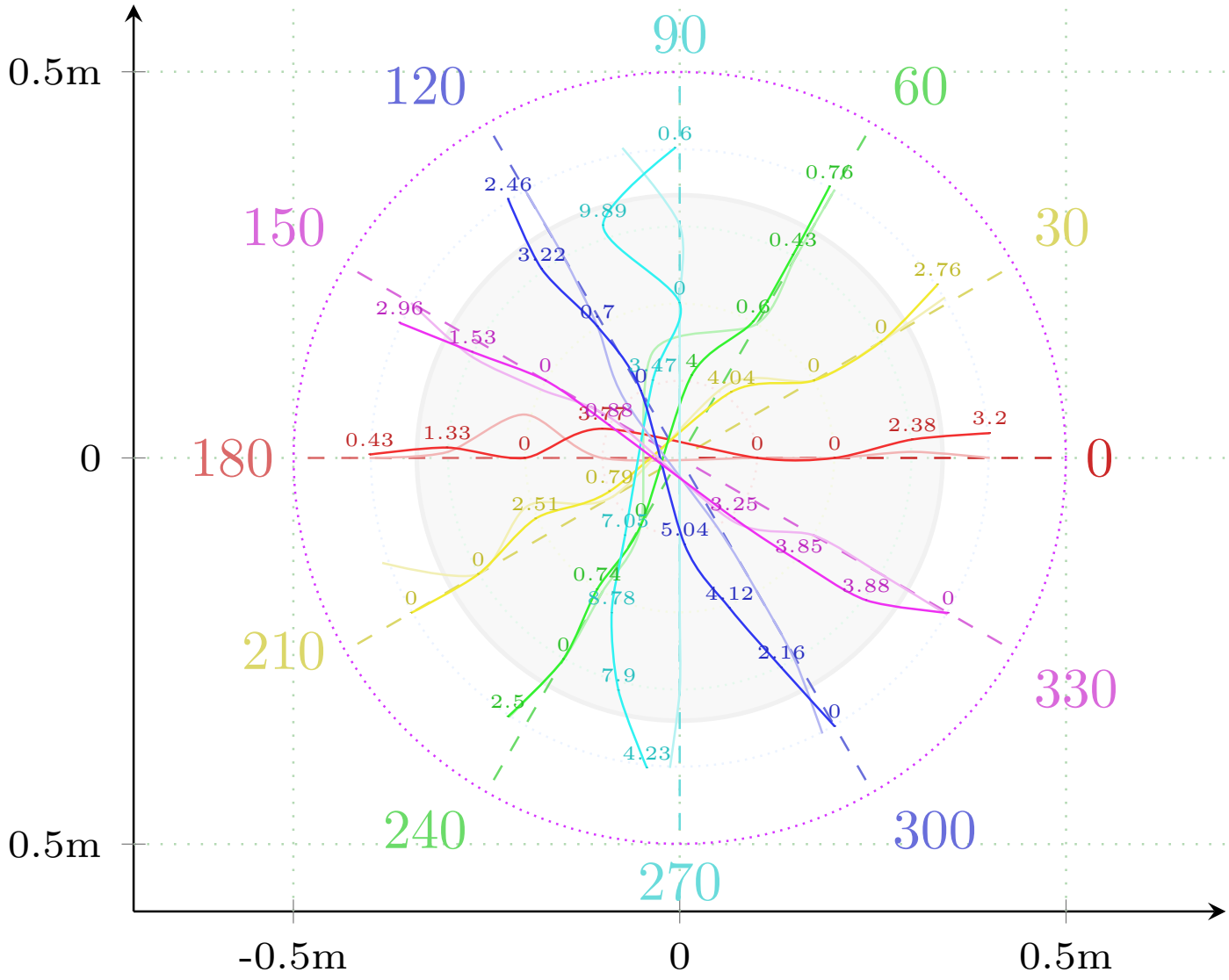


Figure 41: As the previous figure 40, but for targets overshoot instead. These values of the optimized interface are given in percentage for each target.

Target\Angle	0	30	60	90	120	150	Mean	Total
40	0.00	0.83	0.00	7.40	0.00	0.00	**1.37**	**8.23**
30	0.78	0.00	0.00	0.00	0.00	2.07	**0.48**	**2.85**
20	0.00	0.00	0.00	0.00	0.70	1.31	**0.34**	**2.01**
10	0.00	5.10	9.84	0.00	2.41	0.00	**2.89**	**17.35**
-10	0.00	0.00	0.00	0.00	0.29	3.64	**0.66**	**3.93**
-20	5.60	4.44	0.00	0.00	0.00	0.00	**1.67**	**10.04**
-30	0.76	0.00	0.00	0.00	0.18	0.00	**0.16**	**0.94**
-40	0.00	7.46	2.35	1.27	1.84	0.27	**2.20**	**13.19**
Mean	**0.89**	**2.23**	**1.52**	**1.08**	**0.68**	**0.91**	**1.22**	**7.32**
Total	**7.14**	**17.83**	**12.19**	**8.67**	**5.42**	**7.29**	**9.76**	**58.54**

Target\Angle	0	30	60	90	120	150	Mean	Total
40	3.20	2.76	0.76	0.60	2.46	2.96	**2.12**	**12.74** −35.40%
30	2.38	0.00	0.43	9.89	3.22	1.53	**2.91**	**17.45** −83.67%
20	0.00	0.00	0.60	0.00	0.70	0.00	**0.22**	**1.30** 54.62%
10	0.00	4.04	4.00	3.47	0.00	0.88	**2.07**	**12.39** 40.03%
-10	3.77	0.79	0.00	7.05	5.04	3.25	**3.32**	**19.90** −80.25%
-20	0.00	2.51	0.74	8.78	4.12	3.85	**3.33**	**20.00** −49.80%
-30	1.33	0.00	0.00	7.90	2.16	3.88	**2.55**	**15.27** −93.84%
-40	0.43	0.00	2.50	4.23	0.00	0.00	**1.19**	**7.16** 84.22%
Mean	**1.39**	**1.26**	**1.13**	**5.24**	**2.21**	**2.04**	**2.21**	**13.28**
Total	**11.1**	**10.10**	**9.03**	**41.92**	**17.70**	**16.35**	**17.7**	**106.21**
	−35.68%	+76.53%	+34.99%	−79.32%	−69.38%	−44.86%	−44.86%	

Table 4: Mean overshoot percentages between the non-optimized interface, left, and the optimization, right. Besides the accumulated totals, the means (of means) are also shown, since these units are in percentage. As can be seen, there is significant variance across the board in terms of improvement, as indicated by the relative changes in the right table.

8 Conclusion

Rapid progress in computational gains and mobile device technology has sparked a renewed interest in spatial interaction and a surge of innovation that is likely to soon challenge the traditional ways of human-computer interaction. One possible technique for exploiting the space around a mobile device was explored in this work, by the concept of swiping outside the boundaries of a small mobile display.

The main idea pursued was that the interacting space could be defined in a way that optimized the user experience, as formally captured by our hypothesis. To provide a foundation for confirmation or falsification, a solution was developed, implemented and data was collected through experiments, appropriately designed by drawing on lessons learned from existing research. A number of test subjects then interfaced with experimental tasks by completing trials using different spatial interfaces, each conceived on intuitive ideas of how users are likely to perceive the surrounding space.

To define what constitutes a good off-screen interface, a number of properties of the collected trial data were compared against each other. Reducing the interaction duration was assumed to be most relevant and therefore chosen to serve as a primary indicator of an improved user experience. Secondary indicators of overshoot travel and touch count were also covered, to give some insight and measure into how accurate and effortless an interaction proceeds.

What was learned from the data, was that no off-screen interaction could outperform a baseline restricted to on-screen input (with inertia) only, but that an interface based on spherical curvature of the surrounding space came very close. Furthermore, secondary indicators gave the clear impression that such off-screen inputs are without redundancy for all targets within physical reach, while also maintaining full and continuous precision of movement throughout the interaction. These two indicators combined, and taking into consideration that the baseline is significantly advantaged by preexisting user experience, indicates that extending swipes into the surrounding space using such an interface does improve the overall user experience, thereby confirming the hypothesis.

Further attempts were made to bring the solution to a more refined form that could possibly do even better. Through a second experiment, it was confirmed that user inputs do appear spherical, but also that they differ greatly and have properties that are more complex than what can be captured by a simple sphere. A subsequent modification then attempted to conform to these observed properties by introducing compression, rotation and a dynamic location of the mean, based on the collected data. Evaluating this through a third experiment did in fact show a slight performance gain, although with some uncertainty due to a lower number of participants and possible spatial bias.

In all, the work undertaken here has contributed to the body knowledge through three experiments, that combined have illuminated interaction characteristics for when users incorporate the surrounding space into navigational tasks. This was done through a solution for interpreting spatial input that turned out to be on par, if not better, than only relying on-screen input for navigation. This spawns several new research questions, such as whether the solution can be further optimized by incorporating more complex features of the true space. Also, comparing the result to a different approach for spatial modification could be of interest, such as conforming to observed features by minimizing some error through gradients or perhaps parameter sweeping. These questions are left to be explored through future work.

9 Appendices

9.1 Normal vector and offsets

The derivation given below is a variation (and more pedagogical version) of the proof given in section 6.7 of [9], modified for the case of fitting a rectangle.

Deriving the normal vector

To start, we note that the unit-length constraint (4) is in fact quadratic, since

$$\|n\| = \sqrt{n_x^2 + n_y^2} = 1 = n_x^2 + n_y^2 \tag{10}$$

. Also,

$$\lim_{\|\boldsymbol{r}\| \to 0} E_{\text{rect}} \ = 0$$

and hence, the solution to (3) may be obtained by minimizing the length of the residual vector $\boldsymbol{r}$. This is equivalent to minimizing the right-hand side of (5), that is

$$\arg\min_{\boldsymbol{n},\boldsymbol{c}} \|A\boldsymbol{x}\| \tag{11}$$

.

Next, observe from (6) that A is a real-valued matrix that has at least as many rows as columns. In addition, the last two columns for each row of A contains components, that are the result of projecting a motion tracking point onto the spatial plane. Hence, since this has inherent noise, each row of A must be unique. If no single row is reducible to zero as a (non-trivial) linear combination of any other, A then has full row rank. This same reasoning is applicable to the columns of A. So, with full column rank there must exist the decomposition[19] $A = QR$ (where Q is orthogonal and R upper triangular by definition). We may therefore restate the entities we would like to minimize as

$$\begin{aligned}
\|A\boldsymbol{x}\| &= \|\boldsymbol{r}\| \\
&\Updownarrow \\
\|QR\boldsymbol{x}\| &= \|\boldsymbol{r}\| \\
&\Updownarrow \\
\|Q^TQR\boldsymbol{x}\| &= \|Q^T\boldsymbol{r}\| && \text{because Q is orthogonal and therefore leaves the norm invariant} \\
&\Updownarrow \\
\|R\boldsymbol{x}\| &= \|Q^T\boldsymbol{r}\| && \text{since it follows from the orthogonality of Q that its inverse and transpose are equal} \\
&\Updownarrow \\
\|R\boldsymbol{x}\| &= \|\boldsymbol{r}\| && \text{since the inverse of Q is also orthogonal by definition}
\end{aligned}$$

[19] Section 6.5.1 of [9]

in which R, being upper triangular, greatly increases the sparsity of the left-hand side, that is

$$R\boldsymbol{x} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & \dots & r_{16} \\ 0 & r_{12} & r_{23} & \dots & r_{26} \\ 0 & 0 & r_{33} & \dots & r_{36} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & r_{66} \\ 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} c_T \\ c_B \\ c_L \\ c_R \\ n_x \\ n_y \end{bmatrix} \tag{12}$$

. At this point, we could choose to solve for $\boldsymbol{n}$ only by decimating this system to

$$\begin{aligned}
&\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{bmatrix}^T R\boldsymbol{x} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} r_{55} & r_{56} \\ 0 & r_{66} \end{bmatrix} \begin{bmatrix} n_x \\ n_y \end{bmatrix} \\
&= \mathcal{R}\boldsymbol{n}
\end{aligned} \tag{13}$$

, such that we can solve for $\boldsymbol{n}$ by

$$\arg\min_{\boldsymbol{n},\|\boldsymbol{n}\|=1} \|\mathcal{R}\boldsymbol{n}\| \tag{14}$$

. Note, this is in the form of a linear least squares problem (with quadratic constraint, as shown by (10)), a case for which a solution is obtainable using singular value decomposition[20]. That is, we are able to decompose $\mathcal{R}$ as

$$\begin{aligned}
\|\mathcal{R}\boldsymbol{n}\| &= \|U\Sigma V^T\boldsymbol{n}\| \\
&= \|\Sigma V^T\boldsymbol{n}\| && \text{because U is orthogonal}
\end{aligned} \tag{15}$$

and since

$$\lim_{\|\mathcal{R}\boldsymbol{n}\|^2 \to \ 0} \|\mathcal{R}\boldsymbol{n}\| = 0$$

then (14) is equivalent to minimizing the square, giving

$$\arg\min_{\boldsymbol{n},\|\boldsymbol{n}\|=1} \|\Sigma V^T\boldsymbol{n}\|^2 \tag{16}$$

[20] Using the definition in section 6.3 of [9]

. Note that Σ is a diagonal matrix (by definition) with non-strictly decreasing values $(\sigma_1, \sigma_2, \ldots, \sigma_n)$. Furthermore, V^T has no effect on the objective, due to it being orthogonal. With Σ having decreasing eigenvalues, the solution to (16) is then inferable by

$$\begin{aligned} \|\Sigma V^T \boldsymbol{n}\|^2 &= \|\Sigma \boldsymbol{y}\|^2 \\ &= (\sigma_1 y_1)^2 + (\sigma_2 y_2)^2 + \cdots + (\sigma_n y_n)^2 \\ &= \sigma_1^2 y_1^2 + \sigma_2^2 y_2^2 + \cdots + \sigma_n^2 y_n^2 \\ &\geq \sigma_n^2 (y_1^2 + y_2^2 + \cdots + y_n^2) \\ &= \sigma_n^2 \end{aligned} \tag{17}$$

in which the last line holds because

$$y_1^2 + y_2^2 + .. + y_n^2 = \|\boldsymbol{y}\|^2 = \|V^T \boldsymbol{n}\|^2 = \|\boldsymbol{n}\|^2 = 1$$

. Also, the last inequality holds because σ_n is the smallest diagonal value of Σ[21].

Hence, the solution to (16) is found by simply choosing $\boldsymbol{y}$ appropriately as

$$\begin{aligned} \arg \min_{\boldsymbol{y}, \|\boldsymbol{y}\|=1} \|\Sigma \boldsymbol{y}\|^2 &= \|\Sigma \left[0, 0, \ldots, 1\right]^T\|^2 \\ &= \sigma_n^2 \end{aligned} \tag{18}$$

and subsequently solving for $\boldsymbol{n}$ by

$$\begin{aligned} \boldsymbol{y} &= V^T \boldsymbol{n} \\ &\Updownarrow \\ \boldsymbol{n} &= V \boldsymbol{y} \\ &= V \left[0, 0, \ldots, 1\right]^T \end{aligned}$$

which is simply the n-th column of V.

Deriving the offsets

Having found the normal $\boldsymbol{n}$, we are now able to determine the offsets by solving (12) for the zero residuals, i.e. $Rx \approx 0$. We therefore modify and solve (12) for $\boldsymbol{c}$ instead (leaving exactly what we previously discarded in solving for $\boldsymbol{n}$) and get

$$\begin{aligned} 0 &\approx R\boldsymbol{x} \\ &\Updownarrow \\ 0 &\approx R \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{n} \end{bmatrix} \qquad \text{where again } R = Q^T A \Leftrightarrow A = QR \\ &\Downarrow \\ \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} &= \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} & r_{15} & r_{16} \\ 0 & r_{12} & r_{23} & r_{24} & r_{25} & r_{26} \\ 0 & 0 & r_{33} & r_{34} & r_{35} & r_{36} \\ 0 & 0 & 0 & r_{44} & r_{45} & r_{46} \end{bmatrix} \begin{bmatrix} c_T \\ c_B \\ c_L \\ c_R \\ n_x \\ n_y \end{bmatrix} \\ &\Updownarrow \\ \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} &= \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ 0 & r_{12} & r_{23} & r_{24} \\ 0 & 0 & r_{33} & r_{34} \\ 0 & 0 & 0 & r_{44} \end{bmatrix} \begin{bmatrix} c_T \\ c_B \\ c_L \\ c_R \end{bmatrix} + \begin{bmatrix} r_{15} & r_{16} \\ r_{25} & r_{26} \\ r_{35} & r_{36} \\ r_{45} & r_{46} \end{bmatrix} \begin{bmatrix} n_x \\ n_y \end{bmatrix} \\ &\Updownarrow \\ \begin{bmatrix} c_T \\ c_B \\ c_L \\ c_R \end{bmatrix} &= - \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ 0 & r_{12} & r_{23} & r_{24} \\ 0 & 0 & r_{33} & r_{34} \\ 0 & 0 & 0 & r_{44} \end{bmatrix}^{-1} \begin{bmatrix} r_{15} & r_{16} \\ r_{25} & r_{26} \\ r_{35} & r_{36} \\ r_{45} & r_{46} \end{bmatrix} \boldsymbol{n} \end{aligned}$$

which allows us to solve for the offsets directly. With this, both the normal $\boldsymbol{n}$ and offsets $\boldsymbol{c}$ are known, which means we have the equations of all rectangle lines.

[21] This observation of the minimal is implicit from the proof of theorem 6.5 in [9] when changing the problem from maximization to minimization

10 References

[1] Solution protoype. Accessed: 2016-06-25, https://vimeo.com/172508296.

[2] S. Argyropoulos, K. Moustakas, A. A. Karpov, O. Aran, D. Tzovaras, T. Tsakiris, G. Varni, and B. Kwon. Multimodal user interface for the communication of the disabled. Journal on Multimodal User Interfaces, 2(2):105–116, 2008.

[3] P. Baudisch and R. Rosenholtz. Halo: a technique for visualizing off-screen locations. In CHI '03 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pages 481–488, 2003.

[4] businessinsider.com. The downfall of kinect: Why microsoft gave up on its most promising product, 2015. Accessed: 2016-05-09. http://www.businessinsider.com/why-microsoft-xbox-kinect-didnt-take-off-2015-9.

[5] Centre for Sports Engineering Research, Sheffield Hallam University. How The Kinect Works. Accessed: 2016-05-15. http://www.depthbiomechanics.co.uk/?p=100.

[6] Cisco. The Zettabyte Era—Trends and Analysis, 2016. Accessed: 2016-06-10. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/VNI_Hyperconnectivity_WP.html.

[7] A. Cockburn, P. Quinn, C. Gutwin, G. Ramos, and J. Looser. Air pointing: Design and evaluation of spatial target acquisition with and without visual feedback. International Journal Human-Computer Studies, 69(6), 2011.

[8] W. Fang and T. Chang. Calibration in touch-screen systems. Analog Applications Journal, 2007. Accessed: 2016-05-17.

[9] W. Gander, M. Gander, and F. Kwok. Scientific Computing. Springer, 2014.

[10] K. Hasan, D. Ahlström, and P. Irani. Ad-binning: Leveraging around device space for storing, browsing and retrieving mobile device content. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '13, pages 899–908, New York, NY, USA, 2013. ACM.

[11] K. Hasan, D. Ahlström, and P. P. Irani. Comparing direct off-screen pointing, peephole, and flick & pinch interaction for map navigation. In Proceedings of the 3rd ACM Symposium on Spatial User Interaction, SUI '15, pages 99–102, New York, NY, USA, 2015. ACM.

[12] D. Hoiem. How the kinect works, computational photography, cs 498, uiuc. Accessed: 2016-05-06. https://courses.engr.illinois.edu/cs498dh/fa2011/lectures/Lecture%2025%20-%20How%20the%20Kinect%20Works%20-%20CP%20Fall%202011.pdf.

[13] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. A. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. J. Davison, and A. W. Fitzgibbon. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, Santa Barbara, CA, USA, October 16-19, 2011, pages 559–568, 2011.

[14] J. Dischler, Google. Building for the next moment, 2015. Accessed: 2016-06-02. https://adwords.googleblog.com/2015/05/building-for-next-moment.html.

[15] D. Jackle, B. C. Kwon, and D. A. Keim. Off-screen visualization perspectives: Tasks and challenges. In Symposium on Visualization in Data Science (VDS) at IEEE VIS 2015, 2015.

[16] K. Khoshelham. Accuracy analysis of kinect depth data. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XXXVIII-5/W12:133–138, 2011.

[17] S. Mann. Smart clothing: The shift to wearable computing. Commun. ACM, 39(8):23–24, Aug. 1996.

[18] National Institute of Standards and Technology. Special Publication 800-124 Revision 1, 2013.

[19] NaturalPoint. OptiTrack documentation, 2016. Accessed: 2016-05-15. http://optitrack.com/support/faq/general.html.

[20] J. Roth. Personal and Ubiquitous Computing. Springer, 2002.

[21] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern

Recognition, CVPR '11, pages 1297–1304, Washington, DC, USA, 2011. IEEE Computer Society.

[22] A. Shpunt and Z. Zalevsky. Depth-varying light fields for three dimensional sensing, Nov. 1 2011. US Patent 8,050,461.

[23] A. Silberschatz, P. B. Galvin, and G. Gagne. Operating System Concepts. Wiley Publishing, 8th edition, 2008.

[24] D. Vogel and R. Balakrishnan. Distant freehand pointing and clicking on very large, high resolution displays. In UIST '05 Proceedings of the 18th annual ACM symposium on User interface software and technology, pages 33–42, 2005.

[25] windowscentral.com. Discussing backwards compatibility, NXOE and the future with Xbox's Mike Ybarra, 2015. Accessed: 2016-05-03. http://windowscentral.com/exclusive-interview-xbox-one-platform-lead-mike-ybarra.

[26] J.-H. Woo. Wearable device and control thereof, May 1 2016. Pending US Patent Application US20160127624A1.

[27] L. Wroblewski. Mobile First. Jeffrey Zeldman, 2011.

[28] K.-P. Yee. Peephole displays: Pen interaction on spatially aware handheld computers. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '03, pages 1–8, New York, NY, USA, 2003. ACM.

[29] Y. Zhang, J. Zhou, G. Laput, and C. Harrison. Skintrack: Using the body as an electrical waveguide for continuous finger tracking on the skin. In Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, CHI '16, pages 1491–1503, New York, NY, USA, 2016. ACM.