Input Devices for the Next Generation of Computing Platforms

Eric Whitmire

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2019

Reading Committee:

Shwetak N. Patel, Chair

Joshua Smith

Brian Curless

Program Authorized to Offer Degree:
Paul G. Allen School of Computer Science & Engineering

University of Washington

Abstract

Input Devices for the Next Generation of Computing Platforms

Eric Whitmire

Chair of the Supervisory Committee:

Washington Research Foundation Entrepreneurship Endowed Professor Shwetak N. Patel

Paul G. Allen School of Computer Science & Engineering

Our personal computing devices are becoming smaller, more powerful, and more tightly coupled to our body. Emerging computing platforms, such as wearable displays for virtual and augmented reality, promise to revolutionize the way we work, play, and communicate. However, there is a disconnect between the promise of always available, wearable computing and the large, fatiguing, and socially unacceptable movements required to interact with such devices. To enable smaller, more precise interactions, we must leverage systems that track the eyes, hands, and small handheld devices. In this work, I explore the design space of input devices that can be used to enable both high-fidelity immersive experiences and everyday productivity and communication tasks. I present novel sensing techniques to track the eyes with high precision and low power, wearable devices for finger and touch tracking, and techniques for tracking and haptics on handheld controllers.

TABLE OF CONTENTS

ACKNOWLEDGMENTS

First, I thank my family for their unwavering support and for providing me with the resources and foundation that has propelled me to where I am today. Thank you for encouraging my desire to figure out how things work. I also want to thank my amazing partner. Nancy, you've been a constant source of support throughout this journey and I'm grateful to have had you by my side. Pursuing grad school at the same time hasn't always been easy, but it's been a worthwhile journey for us both. I can't wait to tackle our next challenges together.

I want to thank my advisor, Shwetak Patel. Shwetak, you've been a great mentor and I appreciate that you've given me valuable feedback while supporting my independence. The freedom you've given me has allowed me to find and explore my passions.

A big thank you is owed to all the members of the Ubicomp lab, past and present, for providing such a supportive community. I am grateful that I was able to turn to the lab both for critical feedback—sometimes hours of it—and for unquestioned support. As the lab continues to grow and change, I hope you continue to work to maintain the environment that I've benefited so much from. I first want to thank the elders of the lab who were (and are) invaluable mentors. Mayank, thank you for including me in your projects during my first year and teaching me what it means to do research. I enjoyed working with you and learning from you and hope we can continue to collaborate. Keyu, your research was part of what drew me to the Ubicomp Lab in the first place. Thank you for always being willing to lend a friendly ear and for looking out for us.

To those with whom I shared more time during this journey: Alex, you were the first person I met when I visited the lab. Though our research eventually went in separate directions, working late into the night with you on random projects, papers, or puzzles are some of my favorite memories from grad school. You were a great mentor and always gave solid advice, both personally and professionally. I have no doubt that you will accomplish great things and continue to be a valued mentor for future students. Elliot, impressing you was always my standard of a job well done. Thank you for being so selfless with your time and advice. Lilian, thank you for all you've done to make the lab the community it is today. I am grateful of how welcoming you were when I first joined. Despite our proximity, we never had the chance to officially collaborate, but I've always admired the technical rigor you bring to bear on your projects. Edward, I've appreciated our friendship over the years. I admire your ability to craft a vision and build up a team to execute that vision. I look forward to seeing what you and your new lab will accomplish. Mohit, it was great working with you while you were in Seattle. Though the lab wasn't the same after you left, I've enjoyed seeing how your research trajectory has grown. Hanchuan, I enjoyed our collaboration and time together in and out of the lab. Ruth, you brought a sense of fun and laughter to the lab and I admire the breadth of your expertise. Morelle, Manuja, CJ, Matt, and Xin, though we haven't worked closely together, you brought a fresh perspective to the lab and I know the lab is in good hands. Thank you for your friendship and I look forward to seeing amazing things come from you. Finally, Farshid—thank you

for working so closely with me for the last two years and sticking with it even when it felt like nothing would ever work. I've enjoyed succeeding, failing, and learning with you. Several of the projects in this thesis would not have been possible without your significant efforts. I admire your enthusiasm and look forward to seeing what you will build next.

I also thank the undergraduates I've worked with over the years—Divye, Dawn, Andrew, and Michael. You brought new perspectives and made significant contributions to some of the projects in this thesis.

I also thank my friends in the broader CSE community who have provided both personal and academic support. Ravi, you are one of the most selfless people I know and an amazing resource for all things HCI. Kira, it's fitting that we are both starting and ending together. Though our research couldn't be more different, I've enjoyed going through this journey with you. Greg, I've enjoyed collaborating and brainstorming with you and I admire your principled approach to HCI. Kiron, I appreciated the times we would find each other in the hallway and excitedly share demos of our latest work.

I thank the many industry mentors I've have the privilege to work with over the years. Laura Truţoiu, Rob Cavin, Aaron Nicholls, Kenrick Kin, Hrvoje Benko, Christian Holz, Eyal Ofek, Mike Sinclair, Wolf Kienzle, Kaan Akşit, and Michael Stengel. I have thoroughly enjoyed working with each of you, even if just for a few months. Learning from all of you has given me a valuable breadth of perspectives. I especially want to thank Laura Truţoiu for investing in me as a first-year student and for introducing me to the field that eventually became my research focus. Laura, I have learned so much from you over the years and I am forever grateful for the way you have pushed me to grow as an academic and encouraged me in my work. I also want to personally thank Christian Holz. Christian, your feedback has consistently been some of the most insightful and helpful, even when it was blunt. Thank you for your honest advice and I hope we can continue to collaborate.

Finally, I want to express my gratitude to the organizations who have generously funded my research. In particular, I thank the National Defense in Science and Engineering Graduate fellowship, the Adobe Research fellowship, Oculus Research, and the UW Reality Lab.

To those who have supported my dreams, especially Nancy. I couldn't have done this without you.

1 INTRODUCTION

1.1 Overview

Our personal computing devices are becoming smaller, more powerful, and more tightly coupled to our body. The progression from desktop computing devices to devices we hold in our pockets or wear on our bodies has increased the availability of and access to information. However, to date, the shrinking of devices has led to a miniaturization of the display and ways in which we interact with the device. As our touchscreens get smaller, they become more portable and more available, but less useful.

Head-mounted displays (HMDs) for virtual and augmented reality (VR, AR) offer a natural solution to this problem. A near-eye stereoscopic display can achieve a wide field of view and deliver immersive 3D content in a relatively small physical package. These computing platforms offer the potential to revolutionize the way we work, play, create, and communicate.

In driving toward this new computing platform, much attention has been paid to visual output—exploring how to deliver the right photons to the eyes to create compelling 3D experiences. However, to create a tight feedback loop between human and machine, these immersive environments require expressive input. Inherent to the design of head-mounted displays is a notion that the size of the display can and must be decoupled from the way we interact with the device. When our input is no longer coupled to the size of the touchscreens we can fit on the device, it leaves open a rich space for the design of input devices.

Today, the dominant interaction techniques on consumer HMDs use spatial computing. The user wields a handheld controller, whose rotation (3-DoF) or rotation and position (6-DoF) are tracked. Such devices allow the user to reach out and interact with virtual objects in the environment. While such experiences are compelling for gaming applications, which often seek to provide users with a sense of immersion in the gaming environment, this method of input does not scale to productivity applications or the many everyday tasks we currently perform on our personal computing devices. In this thesis, I explore the design space of input devices that can be used to enable both high-fidelity immersive experiences and everyday productivity and communication tasks.

1.2 Designing for Expressiveness, Precision, and Subtlety

In an effort to escape the confines of a small touchscreen, some systems, such as the Microsoft Kinect or Leap Motion, rely on body and hand tracking to provide a *natural user interface*. Other computing systems, such as consumer VR and AR devices, use tracked controllers as a proxy for hand position. These devices are designed and optimized to track the hand and arm in front of the body. Such systems make compelling demos, but prolonged use of such large arm movements lead to fatigue, in what is often referred to as the "gorilla arm syndrome".

Moreover, such experiences are entire unsuitable in many on-the-go scenarios. It is socially unacceptable and at times impossible to uses these kinds of spatial interfaces on a crowded bus, at a restaurant, or while in a meeting. In such situations, it is preferable to use subtle interaction techniques that do not disrupt others or draw unnecessary attention to the user.

Smaller movements improve efficiency, reduce fatigue, and broaden possible usage scenarios. However, such small movements must be accompanied by precision. Imprecise input devices are typically relegated to discrete gesture recognition, e.g., detecting taps on a particular finger or detecting a particular pattern of blinks. On the other hand, continuous, precise tracking enables well-designed gestures and other manipulation techniques that are less constrained by technology limitations. While there are numerous ways to provide input to a wearable computing device, a few classes of input stand out as promising candidates for the small, yet precise input needed on head-mounted displays.

Eye tracking is a useful source of direct and contextual input. Because of the speed of eye movements and little effort required, gaze-based interfaces can seem magical. When designed correctly, such interfaces can be used for on-the-go input and navigation [38]. Because gaze is an indicator of user attention and intention, gaze also provides a useful complementary input. Interfaces designed around a combination of gaze and touch offer compelling glimpses into the possibilities of multimodal input [120].

Fully tracked *handheld objects* can act as tools and provide valuable haptic feedback to the user. Existing optical tracking solutions, such as those found on the Oculus Quest or HTC Vive, require the use of bulky controllers with optical emitters or sensors that cannot be occluded by the hand or body. Commonly used outside-in tracking systems require external infrastructure which limits their use to a particular space. However, for mobile usage, these devices must use inside-out tracking and be usable with the hands at the sides and out of direct line of sight of a head-mounted camera. They must also be small enough to fit in a pocket, which precludes the use of bulky tracking rings.

Specially designed handheld controllers also have a rich potential to provide haptic feedback to the user. However, handheld haptics are traditionally limited to single-point vibrotactile stimulation. More expressive haptic feedback has the potential to improve the speed and accuracy with which we interact with our computing devices [169, 104].

Wearable devices that sense small *finger motions* can enable both direct and indirect manipulation techniques. Such devices are particularly well-suited for augmented reality computing platforms. When tracked in absolute coordinates, designers can repurpose existing objects and surfaces in the environment as interactive elements. For example, a user may reach out and tap a button that appears on their desk surface. Such interactions are less susceptible to fatigue because the user is touching an actual physical surface that supports their arm, instead of reaching out and touching in mid-air. In mobile scenarios, computing systems can make use of indirect manipulation with the arms at the side or even in a pocket. These devices can use spatial finger movements and thumb-to-finger interaction to maximize precision and dexterity while minimizing the overall size of the input.

1.3 Input Beyond Interaction

We typically think of input devices solely for interaction and control. However, as our computing platforms become more tightly coupled to the human body, our input devices also provide a useful source

of information fundamental to the operation of the device itself. Eye tracking, in particular, serves as an enabling technology for a host of features on next-generation near-eye displays. These new displays are increasingly leveraging gaze-contingent features to save power and computation and exert greater control over the user's visual perception.

The human visual system has a wide field of view, but only processes fine detail within a roughly 5° foveal region[50]. Accurate and low-latency [4] eye tracking enables foveated rendering, which saves significant rendering costs by matching the rendering quality with the visual acuity. A high-fidelity image is rendered at the user's central gaze, while a lower-complexity image is rendered at the user's periphery. Foveated displays offer a hardware solution to this problem by using a small high-resolution inset overlayed and blended against a lower resolution backdrop [170].

For some types of displays, eye tracking offers more than just the power savings of gaze-contingent rendering, but is fundamental to the operation of the displays themselves. Recent advances in display technologies have sought to support the eye's accommodation in order to address the vergence-accommodation conflict that plagues many of today's head-mounted displays. Rather than the fixed focal distance found in conventional displays, approaches such as multifocal[60], varifocal[37], or focal surface displays[98] seek to adjust the focal depth of the image to closely match the focal depth of the virtual object. In order to do this, the display either tracks the vergence of the eyes or discerns the object the user is fixated upon and measures its distance from the user. Either approach requires precise eye tracking.

While there are tremendous opportunities enabled by accurate near-eye eye tracking, it remains a challenging problem. Tedious per-session calibration routines, HMD slippage, power consumption, and the integration of additional hardware into an already complex optical pipeline are just some of the challenges that have prevented wider adoption of this technology.

In addition to driving display technologies, understanding the pose and attention of the user enables other valuable behind-the-scenes features. Communication and telepresence is an essential part of a personal computing device. Ironically, by bringing the computing device so close to the body, we have made it more difficult, in some ways, to capture the user. Rendered avatars, driven by an estimated pose, are commonly used to represent the user. An accurate representation of the user's gaze and hand position is essential to creating a realistic and socially present avatar [43]. Understanding the user's visual attention and hand pose can also enable better contextual understanding and activity recognition [154].

1.4 Opening a Design Space of Solutions

While there is significant interest in these sources of input, recent commercialization efforts in this space have focused almost solely on camera-based input. Eye trackers for near-eye displays typically use infrared illumination and an infrared camera to capture the pupil and location on infrared glints on the eye. A common approach for controller tracking is head- or environment-mounted cameras that track the position and orientation of controller-mounted tracking LEDs. Finger tracking relies on either infrared depth cameras or stereo camera pairs.

Camera-based solutions are an incredibly useful sample in the design space of solutions, but focusing solely on this solution has significant limitations. One of the greatest benefits of imaging sensors is also a weakness – they create a lot data. Not only do the sensors themselves consume significant power, but

processing the data obtained from them requires considerable power and computation. On a battery-powered wearable device, such concerns are critical.

Because of the amount of data they generate and the noise and sensitivity levels of image sensors, camera-based solutions are typically limited to under 100 Hz. While sufficient for some applications, other use cases, such as the evaluation of perceptually driven displays, demand higher performance.

For camera-based hand and object tracking solutions, it can be difficult to balance resolution and field of view. For mobile interaction, hand and object tracking must be capable of operating with the hands at the side or in the lap. Such positions make it difficult to achieve consistent line of sight to sensors on a head-mounted display.

Finally, evolving expectations and cultural norms surrounding privacy and wearable cameras must be considered. Outward facing cameras designed to track the hands or handheld controllers will inevitably capture other people. If such a platform is to be widely adopted, the implications of such design decisions must be carefully evaluated.

In this body of work, I explore alternatives to imaging sensors that offer additional flexibility in the design space. Electromagnetic sensing solutions and low-power photosensors are two of the alternative techniques that I focus on in this body of work. These sensing techniques provide lower dimensional analog outputs that more directly relate to the signals of interest. As a result, these techniques offer flexibility and can be customized for solutions that demand either high precision or low power.

1.5 Thesis and Organization

In this thesis, I argue that **input devices for wearable computing platforms can leverage non-imaging sensors to unlock new capabilities and enable more robust, subtle, and precise interaction techniques.**

In support this thesis, the subsequent chapters address the following research questions:

RQ1: *How can we design easily-integrated eye tracking devices that offer low-power and high-precision?*
In Chapter 3, I present the EyeContact system [181], a wearable scleral coil tracker system that enables rapid integration of high-quality eye tracking. Chapter 4 explores a complementary design space of low-power photosensor-based eye tracking.

RQ2: *How can we bring 6-DoF tracking to small handheld objects?*
Chapter 5 presents Aura [180], an inside-out electromagnetic tracking technique for small, handheld VR controllers.

RQ3: *What kind of feedback can we add to input devices to improve the sense of presence in VR?*
In Chapter 6, I discuss alternative haptic feedback techniques and present Haptic Revolver [178], a handheld controller that renders touch, shear, texture, and shape feedback on the finger.

RQ4: *What interaction techniques and hardware can be used to support productive use cases using subtle, finger-based input?*
Chapter 7 explores interaction and text entry based on thumb-to-finger touches through DigiTouch [179].

Finally, Chapter 8, presents AuraBand, an electromagnetic finger tracking device using a ring and wristband form-factor.

2 BACKGROUND AND RELATED WORK

The following sections contain a broad overview of research efforts and trends in near-eye gaze tracking (Section 2.1), tracking techniques for handheld controllers (Section 2.2), haptic feedback techniques on handheld controllers (Section 2.3), and finger and touch tracking techniques (Section 2.4). Because several of the projects in this thesis rely heavily on electromagnetic tracking techniques, an additional section reviews technical advances in electromagnetic tracking independent of the domain of input devices (Section 2.5).

2.1 Eye Tracking

2.1.1 Optical Imaging Eye Tracking

Video oculography is the most widely used method for eye tracking. Most video-based eye trackers rely on infrared illumination of eye and an infrared-sensitive video camera that detects either the location of the pupil or glints on the cornea. To avoid blocking the user's vision, an IR-selective mirror, or "hot-mirror", can be used to reflect the infrared illumination out of the user's field of view. A calibration procedure is used to construct a mapping between glint/pupil locations and eye orientation.

A high end commercial video eye tracking system, such as the SR Research EyeLink 1000 Plus [132], is capable of sampling at $1000\,\text{Hz}$ with 0.33° average accuracy. Commercial wearable eye tracking systems from Tobii and SMI, take the form of lightweight glasses with IR illumination. These systems often have an accuracy on the order of 0.5° with an output data rate of $60\,\text{Hz}$ to $100\,\text{Hz}$ but are not designed to be compatible with an HMD. Dual Purkinje tracking is an alternative high precision optical tracking technique, but it requires a complex set of mirrors and servo motors that make it difficult to implement in head-mounted displays. For a more thorough review of wearable video-based eye trackers, see the review by Bulling and colleagues [16].

Video-based eye tracking solutions [40, 29, 62, 67] have recently been adapted for use in an HMD. For example, the SMI tracking system is available as an add-on package for the Oculus Rift DK2 and Samsung Gear VR [62]. It achieves 0.5° to 1° accuracy with a data rate of $60\,\text{Hz}$. Additional solutions from Arrington Research and Tobii offer similar performance. Although these solutions will likely improve with time, current solutions suffer from a low data rate and accuracy and are designed specifically for the optics of a particular HMD.

2.1.2 Electrooculography

Another method for eye-tracking is electrooculography (EOG). The eye is the source of an electric dipole between the cornea and the retina; this corneoretinal potential can be up to 1 mV, and varies as a function

of eye orientation. Electrooculography estimates eye movement by measuring this corneoretinal potential through electrodes placed around the eye. These systems generally offer a much higher sampling rate than video-based systems, and processing the resulting data is much simpler. Although the EOG signal is not rich enough to enable accurate eye tracking, it can be used to detect blinks and eye gestures. For example, Bulling and colleagues demonstrated wearable EOG goggles capable of detecting a set of eye gestures [16].

2.1.3 Scleral Coil Tracking

Scleral coil tracking is the gold standard for eye tracking, particularly within the medical community. It offers a sampling rate often approaching $10\,\mathrm{kHz}$ with an accuracy better than 0.1°. This technique was developed by Robinson in 1963 [138] and refined by Collewijn and colleagues in 1975 [27]. The subject wears a wire coil embedded within a silicone ring that sits on the sclera of the eye. A thin wire lead is connected to an external measurement unit. When placed in a magnetic field, a voltage is induced in the coils, which can be amplified and measured.

A challenging aspect of this technique is the generation of the magnetic fields. Three orthogonal magnetic fields, either at separate frequencies or in phase quadrature, are used to induce three separable signals in the scleral coil. In order to ensure a uniform field, Helmholtz coil pairs, often several meters long, are used to generate the magnetic fields. Even so, the uniform region is small and the use of head restraints is often required. The hardware imposes significant space and cost constraints, preventing the adoption of this technique outside of dedicated facilities in medical centers. Today, scleral coil eye tracking is used for the diagnosis of vestibular disorders.

2.1.4 Improvements to Scleral Coil Tracking

In an attempt to increase the usability of scleral search coil methods, several projects have attempted to eliminate the wire connecting the scleral coil to the measurement device. Reulen and Bakker describe a double magnetic induction (DMI) technique [133] in which a short-circuited scleral coil is used. By placing the subject within an AC magnetic field generated by a set of Helmholtz coils, a voltage is induced in the scleral coil loop. Current within this loop creates a secondary magnetic field which can be detected by a set of coils directly in front of the eye. Similar work by Bremen and colleagues [15] showed a wearable version of the DMI technique. Fundamentally, DMI suffers from a weak signal to noise ratio because of the need to measure the secondary magnetic field. This technique also limits the number of magnetic fields used and is sensitive to the mechanical orientation of the field and detector coils.

Roberts and colleagues showed a wireless scleral coil system that operated on a similar principle [137]. By shorting the scleral coil with a capacitor embedded in the silicon annulus, they created a resonant circuit. Using a series of high frequency (5 MHz) pulses from a nearby transmitter coil, oscillations were induced in the scleral coil loop. Several receiver coils measured the decaying signals. The relative strength of the signal in the receiver coils determines the orientation of eye gaze.

Thomassen and colleagues noted that the small uniform magnetic field region of many trackers made it difficult to carry out head-unrestrained experiments [162]. The authors adapted the tracking algorithm of an off-the-shelf scleral coil tracking system to correct its performance when the head moves away from

the uniform region, though this technique still requires significant instrumentation of the environment. Plotkin and colleagues have also attempted to address the nonuniformity problem with a planar transmitter placed in a fixed location behind the user [124]. Their system can estimate the position and gaze orientation of a scleral coil. Although their design is portable, it is too large to be mounted on the user and does not allow the user to move more than 10 cm during an experiment.

2.1.5 Low-power Eye Tracking

While scleral search coil methods represent the gold standard of eye tracking, other work has sought to track gaze with a minimal amount of power. In fact, early efforts to track eyes in the 1950's through 1970's relied on limbus tracking methods [142] that use infrared illumination and photosensors to estimate the horizontal and vertical direction of gaze. These systems would use multiple sensors to track the boundary between the iris and sclera. Unfortunately, these systems were difficult to calibrate and relied on a linearity in the difference between two photosensor outputs. As imaging sensors became more readily available, this tracking technique fell out of favor.

However, recent interest in wearable, low-power eye tracking techniques has caused a renewed interest in photosensor tracking methods that rely on limbus or specular reflection tracking. A line of work by Rigas et al. simulates the effects of photosensor placement [136] and explores hybrid approaches to solve sensor slippage [135]. Unfortunately, these efforts are currently limited to simulation. Li et al. designed LiGaze[86], an attachment for a VR headset that uses 16 photodiodes around the display to measure the reflected screen light and additional photodiodes on the back side to measure screen brightness. They achieve 6.1° accuracy while consuming only $791\,\mu\mathrm{W}$. Followup work adds custom LED illumination to eliminate the dependence on the display, although the angular gaze error is not evaluated [87]. The Eye Touch system uses four IR LEDs and four IR sensors per eye [166] and was eventually upgraded to six LEDs and sensors per eye [164]. However, the performance of these systems was not thoroughly evaluated. Followup work showed that this system achieves 0.93° error [165], though this appears to be subject to overfitting issues. AidedEyes[64] doesn't attempt full gaze tracking, but uses an LED and phototransistors to track eye activity. Because infrared illumination and a photosensor operate as a sort of proximity sensor, other researchers have developed techniques to use this sensing modality to track gaze with the eye closed [28]. Other systems use scanning lasers and photosensors to track the eye [63, 2].

Another approach to low-power gaze sensing relies on wearable cameras with either low resolution or low sampling rates. iShadow[99] achieves 3° error running at $70\,\mathrm{mW}$ using a single camera and sparse subsampling of pixels. Mayberry et al. substantially improved this design in CIDER[100], which achieves 0.6° error using only $7\,\mathrm{mW}$ at $4\,\mathrm{Hz}$. Rostaminia et al. followed up on this work with iLid, which detects eye closures and blinks[141]. InvisibleEye, by Tonsen et al used multiple millimeter-size cameras to track gaze [163]. They achieve 1.79° average error on their most realistic dataset using three 5x5 pixel images. Damian et al. used an event camera in a "silicon retina" system to track edges of eye features [34].

These prior works feature many exciting avenues for research, but many do not implement a full system or evaluation.

2.2 Handheld Controllers

2.2.1 Inside-out Tracking Techniques for Handheld Controllers

While there are many commercial and research techniques for tracking handheld objects by instrumenting the environment, this review is limited to inside-out approaches that are compatible with mobile use. Commercial devices such as the Windows Mixed Reality system and Oculus Quest use head-mounted cameras that track a controller with an external LED ring. While this technique is precise, it requires the use of bulky controllers and line of sight to the headset, preventing use when the hands are not visible to the headset. These devices often use multiple cameras to maximize controller tracking voluem. The upcoming Vive Focus uses ultrasound and IMU tracking, though there are few publicly available details about how this works. Magic Leap has recently released a proprietary electromagnetic tracking solution, with a multi-frequency transmitter in the controller and a receiver coil in the headset.

In the research space, a number of different techniques have been explored to eliminate the use of bulky tracking rings and markers. Pocket6 is a solution that uses ARKit on an iPhone X to localize the controller using the built-in SLAM and IMU fusion algorithms [9]. Pandey et al. demonstrate a technique to track markerless controllers using only the front-facing camera on the HMD [113]. These classes of solution tend to be power-hungry.

Though not explicitly a controller tracking technique, Shen et al. presented a technique [146] to track the position of a smartwatch using IMU sensors and a kinematic model. Ultrasound tracking systems have long been used for tracking the head [35, 41, 172]. These rely on pairs of beacons and microphones that use time-of-flight measurements to estimate 6-DoF pose. Though they are usually small and light, they are sensitive to occlusion and ultrasound noise. Nandakumar et al. use RF backscatter to track the position of sub-centimeter scale devices [106]. Although they can localize objects tens of meters away, their accuracy is not sufficient for VR controller trackers and requires instrumenting the environment. See Baillot et al. for an additional review of VR tracking technologies [10].

2.3 Haptic Feedback on VR Controllers

The small form factor, low cost, and low power of vibrotactile actuators have led to their dominance in commercial VR controllers. The positionally tracked controllers offered by consumer VR systems (e.g. Oculus Touch or HTC VIVE controllers) include customizable vibrotactile feedback. However, the amount of information that can be conveyed by vibrotactile stimulation is limited and its usage is typically limited to simple touch events or notifications. Some research efforts have investigated how to use vibrotactile stimulation to render surface textures with a particular focus on how stimulation parameters impact users' perception of a surface [30, 156].

Recent academic and commercial efforts have attempted to move beyond vibrotactile feedback. For example, Benko et al. demonstrated NormalTouch and TextureTouch, controllers that render normal forces on the fingertip [12]. Other efforts have focused on using controllers to render the sensation of holding objects [155, 25, 24, 84]. Zenner and Krüger presented Shifty, a handheld device that shifts its weight distribution to simulate holding objects of different weights [194]. Tactical Haptics designed a controller

that simulates friction forces in the palm due to holding an object using sliding tactors in the device handle [129]. By moving these tactors on opposite directions, the controller can simulate torsional forces as well.

2.3.1 Wearable Haptic Devices

In addition to handheld haptic controllers, there are several options for wearable haptic devices. Glove-based exoskeletons such as the Exos [39], Dexmo [36], CyberGrasp [31], and Rutgers Master II [14] all use actuators at the fingers to resist grasping forces. Though these devices have the advantage of grounding at the wrist, they tend to be bulky and require a nontrivial amount of setup compared to a handheld controller. Such devices are also unable to render shear forces and motion underneath the fingertip. In contrast, our device can render the sensation of sliding across a virtual surface.

Finger-mounted haptic devices are another common form factor. These are often strapped or clipped onto one or more fingers to provide fingertip sensations as the hand moves. In recent years, researchers have explored rendering contact [147, 117], pressure [128], tilt [23, 110], and shear forces [111, 145] with these devices. Yem and Kajimoto developed the FinGAR device, which uses a DC motor and electrodes to provide skin deformation and vibration [189]. Wolverine is a four-finger device that uses braking to simulate grasping rigid objects [24]. Go Touch VR designed a device that clips onto three fingers and renders contact and pressure as a user grasps a virtual object [47]. Like the exoskeleton devices, these devices require careful mounting to the fingers before use. Moreover, with our device, one can simply pick it up and begin using it without having to strap anything to the fingers. However, unlike most controllers, including our device, these finger-mounted devices do not restrict hand posture during use. We note that there are many more examples of finger-mounted haptic devices. For a more comprehensive review of such devices, we direct the reader to Pacchierotti et al. [112].

2.3.2 Desktop Haptic Devices

In addition to handheld and wearable devices, there are a number of efforts exploring haptics using environmentally grounded systems. Robotic arm actuators such as the PHANToM [97], Haptic Master [168], Novint Falcon [109], and Haption Virtuose [51] excel at rendering larger, externally grounded forces against the finger or hand. These are often used in applications such as tele-operated surgery, 3D sculpting, gaming, and interactive training. Recently Araujo et al.'s Snake Charmer [6] used a robotic arm with custom attachments to render various surface features on demand. The Touch Thimble is another example of combining attachments with a robotic arm for haptic feedback [78]. In this work, a spring loaded thimble keeps the touch surface suspended from the fingertip until a virtual surface is contacted.

Azmandian et al. showed how retargeting can be used with passive proxies [8] to reuse the same proxy for multiple virtual objects.

Other desktop devices render fingertip sensations during stationary use [188, 171, 182, 77] or in a limited range [17]. For example, the Plank is a desktop haptic device that uses a spinning wheel to render friction and various terrain shapes [171]. Other haptic devices use the tilt of a platform under the finger to convey surface information [182, 77].

2.4 Wearable Devices for Dextrous Input

Research into input techniques using the fingers has explored ways to provide more expressive input on tiny devices [184, 69], to leverage the human body as an input surface [197, 53, 61, 179], and to use freeform hand tracking as input [175, 160]. The following sections outline the most relevant work on thumb-to-finger input, finger input using magnetic sensing, glove-based devices, and other techniques for tracking and text entry.

2.4.1 Thumb-to-finger input

Due to the anatomy of the hand, touching the thumb to the fingers is a natural and expressive interaction. This interaction benefits from both tactile and proprioceptive (an innate sense of the body's position and movement) feedback. Prior work [61, 127] has explored this style of thumb-to-finger interaction for various applications. For example, *DigiTap* [127] uses a wrist-mounted accelerometer and camera to detect thumb-to-finger taps. The accelerometer detects when a tap occurs, and awakens the camera to observe where the tap occurred. It can identify discrete taps on 12 locations (three regions per finger).

Other projects explore subsets of thumb-to-finger input, e.g. *NailO* [66], *Ringteraction* [45], and *FingerPad* [19], which place small sensors on the finger to enable thumb gestures. In these systems, the interaction surface is limited to a small portion of a single finger.

Saponas et al. [143] demonstrated a forearm-based electromyography device that can classify a set of hand gestures, including thumb-to-finger gestures. However, the discrete nature of the classification makes it unsuitable for fine-grained thumb-to-finger sensing.

To estimate thumb and finger positions, researchers have also explored vision-based techniques [18, 70]. *CyclopsRing* [18] uses a unique fisheye camera placed between the index and middle finger, to distinguish between a number of hand gestures, including several thumb-to-finger touches. However, it is unclear how accurately it can detect touch events and estimate the position of the thumb along the finger, due to occlusion issues. *Digits* [70] uses a wrist-mounted infrared camera to reconstruct a 3D model of the hand and fingers.

These prior works demonstrate the interest in and potential of a form of subtle thumb-to-finger interaction. Looking at the field as a whole, however, the challenge in this space remains expressivity without compromising on form factor. Most of these devices are used for discrete gestures or button-based input. Many also struggle at detecting contact events. While useful in some cases, for broader adoption, there is a need for more precise *tracking* of fingers using wearable devices and more precise detection of *contact* events for robust interaction.

2.4.2 Magnetic Sensing Approaches

Some work has adopted the use of magnetic sensors and a passive magnet glued to a fingertip to enable more precise and continuous fingertip sensing. Abracadabra[52] used this technique for selection and coarse cursor control on a smartwatch. The *DigitSpace* [61] prototype detects the position of a magnet attached to the thumb along the length of a finger using a chain of Hall effect sensors. In 2016, Lyons

showed a smartphone-based magnetometer tracking system that enabled 2D interaction on the side of a cardboard VR device using a permanent magnet [89]. SynchroWatch[134] used a thumb-mounted magnet to detect a number of oscillating thumb gestures by tracking the relative motion of the thumb over time. Other examples include Nenya [7], which tracks the 1D rotation of a ring, FingerPad [19], which tracks subtle motions of the thumb against a finger, and uTrack [20], which tracks the position of the thumb against the fingers.

Magnetic tracking systems that rely on permanent magnets have size and complexity advantages because they can rely on off-the-shelf magnets and magnetometers. However, this comes at the cost of precision. Magnetometers are generally limited in their accuracy and sampling rate and such systems must contend with interference from the Earth's geomagnetic field. As a result, the operating range of such systems is generally limited to several centimeters. Prior work that uses finger-mounted magnets for continuous tracking, such as uTrack [20] and FingerPad [19], place both the sensors and the magnets on the fingers themselves.

In contrast, AC magnetic tracking relies on one or more oscillating magnetic fields.

Finexus [21] used finger-mounted electromagnets and to track the fingers with respect to a base station with four magnetometers. However, due to the use hall-effect magnetometers, the range of this device was still insufficient for a wrist-mounted sensor.

These efforts demonstrate promising interaction techniques that can be achieved with more fine-grained tracking, but there is still a need for a solution that operates independent of the geomagnetic field and only requires a single point of instrumentation on the finger.

2.4.3 Glove-based input

There are several examples of glove-based interfaces that have been proposed for input and text entry. Miller *et al.* [103] created a glove with a 2D input surface along the length of the fingers using an array of conductive threads, but only demonstrated the ability to perform simple targeting tasks. Though not a full glove, *Plex* [192] is a wearable finger covering that uses piezoresistive fabric to enable thumb-to-finger touches. Similarly, *TIMMi* [191] extends this to enable the reconstruction of finger bend and touch pressure.

A few commercial data gloves [1, 159, 81] attempt full hand pose reconstruction using bend sensors in the finger joints. Theoretically, the necessary thumb and finger positions can be extracted from this data, but they are not accurate enough to reliably detect taps or estimate the relative position of the thumb and finger. Other commercial gloves, such as Peregrine Glove[46] uses a set of touch sensitive regions along the finger to lay out arbitrary buttons. However, they suffer from lack of tactile feedback due to the thickness of the glove and the touch sensitive coil.

KITTY [79] is a glove that combines four contacts per thumb with six contacts on four fingers, to offer 48 button combinations, suitable for text entry. This results in an expressive, but complex set of possible touch points. By placing a continuous input space along each finger, we enable a split-QWERTY keyboard with a familiar layout. *Argot* [119] is a one-handed glove with 15 buttons, enabling text-entry using multi-tap and a T9 dictionary. Rosenberg and Slater [140] proposed a chording-based glove with seven buttons.

With training, chording-based techniques can achieve high text entry speed (up 16.8 wpm after 10 hours [140]), but are difficult to master.

2.4.4 Smart Rings

Smart rings provide an attractive form-factor for always-available subtle input. Commercial smart rings offer features like fitness tracking, heart-rate tracking, inertial gesture sensing, and NFC payments. Rings have also been shown useful for subtle input [7], interaction with 2D surfaces [187, 105, 69]. LightRing [69] uses a gyroscope and IR proximity sensor to track finger motion on a 2D surface.

2.4.5 Other Approaches for Finger Tracking

Researchers have leveraged other sensing techniques to estimate fine-grained finger motion. Kim et al used infrared illumination from the wrist to track hand pose [70]. Project Soli [176] uses radio frequency signals to recognize subtle finger gestures. Pyro uses the pyroelectric effect to recognize small thumb gestures [48]. Bioimpedance has been used to reconstruct the cross-sectional impedance of the arm and recognize various hand gestures [195, 196].

Other systems use sensors on the opposing arm or in the environment to track finger motion. FingerIO uses sonar from a smartphone to measure the 2D position of the fingertip [107]. SkinTrack uses a ring-smartwatch pair that leverages electrical waveguides to track the 2D position of the finger on the opposing arm [197].

Still other systems use finger strips to estimate finger and hand pose [192, 191, 59, 61, 5, 46, 1, 159]. These systems require significant augmentation and are less appropriate for everyday usage.

Finally, camera-based approaches are a common approach to hand- and finger- tracking. These commonly rely on infrared depth sensing [160] or markers on the hand [175]. Unique placements of the cameras have lead to interactive systems that offer subtle input [88, 70, 18].

2.4.6 Other Text Entry Methods

Other systems for eyes-free text entry for wearables systems include vision-based approaches [95, 153, 173] and approaches using external devices [49, 115, 90]. Sridhar et al. [153], *Vulture* [95], and *PalmType* [173] use vision-based hand-tracking for text entry. Sridhar et al. use a mid-air chording technique and achieve 22.2 wpm, with participants entering a word until they reach their peak performance. *Vulture* is a mid-air word gesture keyboard that achieved 20.6 wpm at the end of 10-sessions, while *PalmType* uses the index finger of one hand to type on the palm of other hand, and achieved 4.6 wpm. Vulture and PalmType were prototyped using the Vicon motion tracking system, as they require fine-grained hand-tracking. These systems suffer from occlusion issues and in-air typing results in fatigue after extended use. *TiltType* [115] uses controlled tilting of the wrist for input, which can be fatiguing for the user, and *TypingOnGlass* [49] uses swiping on the frame of a Google Glass for text entry (8.7 wpm), which attracts attention and is

socially awkward [167]. Twiddler[1] is a hand-held device for text entry using multifinger chording. Prior work has shown participants achieve 26.2 wpm after 400 minutes of practice.

2.5 Advances in Electromagnetic Tracking

AC electromagnetic tracking has a rich history of enabling precise, 6-DoF tracking [131, 80]. Since its inception in the late 1970's, the technique has been used for surgical tracking [82, 42, 185], biomechanical analysis [118, 102], virtual reality tracking [73], localization [22], and HCI [122, 21].

In this approach, an electromagnet is driving at a particular frequency and signal processing techniques are used to filter out anything but this frequency on the receiver (including the geomagnetic field). A typical electromagnetic tracking system contains a transmitter base station with a three-axis orthogonal coil and one or more three-axis receivers, typically realized using orthogonal coils. The oscillating magnetic flux through the coils induce an oscillating voltage of the same frequency which can be amplified and measured. Filtering or synchronous detection is used to isolate the field of interest from other ambient magnetic fields, including the geomagnetic field. Original systems relied on iterative approaches to estimate the pose of the tracked objects, however, recent techniques have explored closed-form or closed-loop solutions that improve performance [73, 44]. There have been attempts to simplify the computation involved in electromagnetic tracking by utilizing rotating transmitters [148, 44] or fewer transmitter coils [114, 33, 32]. Magnetic field distortions when tracking near metallic objects is a common drawback of AC magnetic tracking, particularly when used in dynamic environments. Some work has attempted to account for magnetic interference through a secondary calibration step [74, 75].

Some electromagnetic tracking solutions rely on magnetometers instead of field coils. Dai et al. demonstrate an electromagnetic tracking technique using a single transmitter coil and a 3-axis magnetometer [33, 32]. In Finexus, Chen et al. use four magnetometers to track the position of eletromagnets placed on the fingertips [21]. Both of these approaches are limited to shorter distances (<20 cm). Islam et al. show a technique using resonance coupling to improve the efficiency of electromagnetic tracking systems [65].

A few groups have build electromagnetic tracking systems suitable for wearable use. Pirkl et al. developed a wearable low-power electromagnetic system, although due to complexities in positional tracking, used it only for gesture recognition. Roetenberg et al. created a wearable EM tracking system with $5\,\mathrm{mm}$ accuracy [139] using a pyramidal structure of transmitter coils. For a more thorough review of magnetic positioning systems, see [116].

In the commercial space, electromagnetic tracking is performed by products from Polhemus, Ascension Technology (now NDI), and Sixense. These products consist of a large transmitter that emits a consistent magnetic field that spans a volume on the order of a cubic meter. The base stations are coupled with small receiver sensors that plug into a usually large processing hub.

In general, such devices offer incredible tracking precision and accuracy, but rely on large infrastructure and distortion-free fields that make integration into a mobile device difficult. Magic Leap has recently released a mobile EM tracking system for their AR controller. While their system is proprietary, the relatively large controller and placement of the sensor suggest this system also relies on distortion-free dipole

[1] http://twiddler.tekgear.com/

fields. There remains a need for head-mounted inside-out electromagnetic tracking solutions that do not require a particular field structure or large transmit/receiver coils.

3 EYECONTACT

3.1 Introduction

Accurate and high-speed eye tracking is important for enabling key scenarios in virtual reality (VR) and augmented reality (AR). Eye tracking could enable a new class of gaze-mediated input [72] and techniques such as foveated rendering [50], which can reduce the computational demands of AR/VR by focusing render quality at the user's gaze location. Virtual avatars could be made more realistic by including eye tracking information [43], which is impossible to measure with traditional motion capture systems while wearing an head-mounted display (HMD). High-accuracy eye tracking could also enable studies of how the human vestibulo-ocular system responds to virtual reality.

Existing research on wearable eye tracking systems has focused predominantly on the use and improvement of optical tracking techniques. However, the gold standard for high resolution eye tracking is still magnetic tracking with scleral search coils (SSC) [58]. Scleral coil tracking can record small amplitude motions with high temporal ($> 1\,\mathrm{kHz}$) and spatial resolution (calibrated error $< 0.1^{\circ}$). In this technique, the head is positioned between large Helmholtz coils, which generate a uniform magnetic field. A wire loop embedded in a silicon annulus, shown in Figure 3.2, is placed on the sclera of the eye. The magnetic field induces a voltage in the scleral coil according to its orientation [138]. By examining the magnitude of the voltages induced in the thin wires leading from the coil, the system estimates the eye's orientation.

One of the major limitations of SSC tracking is the need for large generator coils several meters in diameter or a head restraint such as a bite bar or chin rest [162]. To overcome these limitations, we propose a wearable scleral search coil tracking system compatible with an HMD, such as those used in virtual reality systems. By mounting smaller generator coils directly on the HMD, as shown in Figure 3.1, we constrain the position of the coils relative to the head, allowing the subject to move freely and eliminating the need for a stationary head or room-sized coils.

The insertion of a scleral search coil is an inherently invasive procedure, typically done with a topical anesthetic, and requires supervision by a trained technician. Hence, we do not recommend the use of a scleral coil tracker for consumer use in VR/AR systems. We instead envision researchers using it when they need eye tracking with high accuracy and high temporal and spatial resolution for wearable HMD scenarios. For example, a psychophysics researcher may use our system to obtain high-resolution data to study microsaccades, small (0.2°) eye movements that occur during fixation [96], while presenting visual targets on the HMD to better understand eye behavior variations. Existing video-based eye trackers for HMDs often have insufficient accuracy and temporal resolution for these kinds of studies. We also hope to encourage researchers to consider our SSC system to obtain ground truth data for evaluating more traditional optical eye tracking systems. An additional possible use for our system is in the medical field, where scleral coil tracking is the gold standard for the diagnosis of subtle vestibular, ophthalmological, and neurological disorders [58].

Figure 3.1: The EyeContact scleral coil tracker can clip to an HMD and does not require the use of a head mount or room-sized field coils

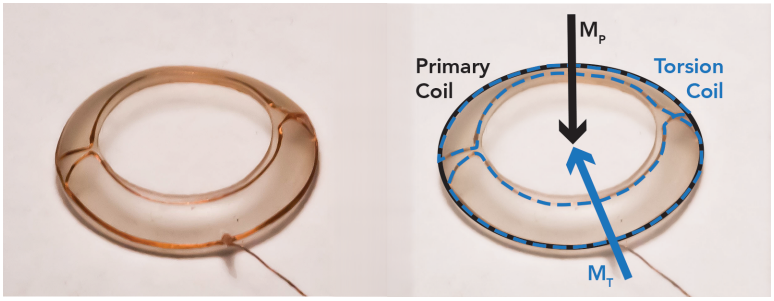

Figure 3.2: 3D torsional scleral coil from Chronos Vision. The primary coil captures flux flowing through the contact while the torsional coil captures flux traveling across the contact.

The specific contributions of EyeContact are:

1. a mobile, head-mounted system for high-speed and high-accuracy eye tracking that does not require instrumentation of the external environment,
2. a unique coil placement that enables reconstruction of gaze and the position of the scleral coil in space,
3. a calibration technique that accounts for the diverging fields created by the smaller generator coils,
4. a highly accurate mechanical test rig with five degrees of freedom that allows us to thoroughly characterize the magnetic field and enables an evaluation of the accuracy and precision of a SSC tracking system independent of the fixation accuracy of a user.

3.2 System Design

We present EyeContact, a scleral coil tracking system designed for use with virtual reality headsets. We use five small coils as shown in Figure 3.4 (left), and rigidly mount the coils on the user's head, constraining the magnetic flux orientation to the user's head orientation. Each coil oscillates at a unique frequency and creates a unique magnetic field. A key challenge in our system is that, in contrast to Helmholtz coils, these coils generate diverging fields and the field orientation strongly varies with position, as shown in Figure 3.3 (right). Further complicating our system is the fact that a scleral coil translates in space along the surface of the eye as the eye rotates. As it moves relative to the field coils, the magnitude and direction of the magnetic fields at the scleral coil change. However, since the decomposed five magnetic fields are unique at each position in space, gaze estimation is still possible. Because the fields vary with eye position, we are also able to recover the positional offset between the tracker and the scleral coil. This allows us to account for any shifts or slippage of the HMD on the user's face.

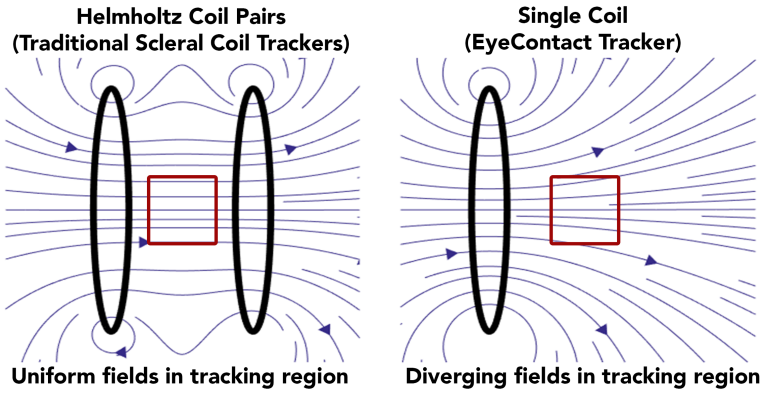

Figure 3.3: (left) Traditional scleral coil trackers generate fields using Helmholtz coil pairs, which produce a small volume where the magnetic field is insensitive to small head movements. (Right) Fields generated by the unpaired coils in our system create a diverging magnetic field.

The use of five generator coils in this arrangement ensures that three coils are close to each eye and can provide a reliable signal (the central generator coil is shared by both eyes). Although these three generator coils alone are sufficient for reconstructing the orientation of the eye, the two coils furthest from each eye provide a weak signal that further improves the accuracy of the gaze estimation. The size of the coils was optimized to balance magnetic field strength and the weight of the tracking device. By placing the coils close to the eyes, we can generate magnetic fields with comparable strength to traditional field coils while using much less current (less than 1 A per coil).

For the scleral coil, we use a 3D scleral contact (which contains a torsional coil) from Chronos Vision. In contrast to previous efforts, we use the torsional coil for more than just torsion estimation. Rather, we consider the entire scleral coil as a biaxial magnetic field sensor that measures two of the three components of the magnetic field.

Each generator coil consists of 50 turns of 26 AWG magnet wire wound around a circular 3D printed ABS

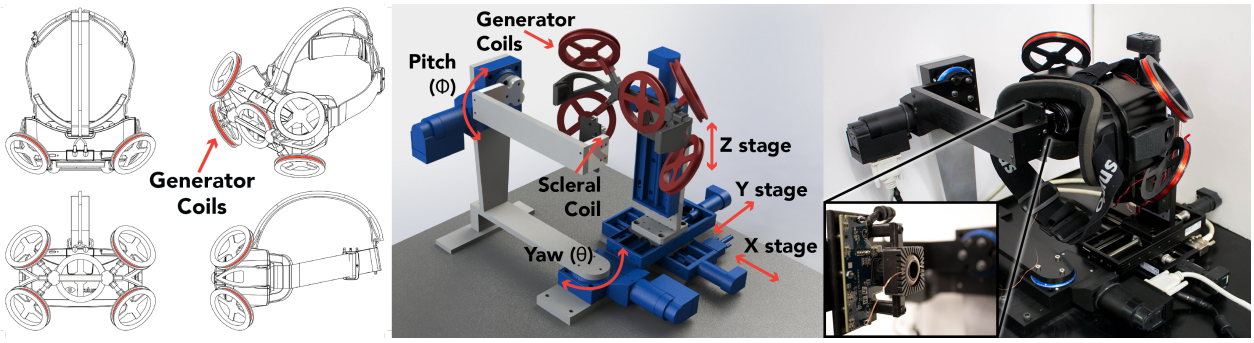

Figure 3.4: (left) The scleral coil tracker consists of five field coils arranged around the HMD. It is designed to snap on to the Oculus Rift DK2, but it can be easily adapted for other HMDs. (center, right) The 5-DOF mechanical evaluation rig allows us to control the scleral coil orientation (pitch, yaw) and adjust the position of the HMD/tracker with respect to the scleral coil. (right, inset) The scleral coil rests in the arm of the test rig.

frame, $8\,\mathrm{cm}$ in diameter. The coils fit together and form an attachment for an HMD. Our implementation is designed to fit on an Oculus Rift DK2, but could easily be modified to fit other VR devices. A system diagram is shown in Figure 3.5. The coils are driven by sinusoidal currents synthesized from a desktop computer. By keeping the frequency of the coil stimulus in the usable audio range, commodity audio hardware can be used for synthesis and amplification. We use a desktop computer running Max 7 software to synthesize sinusoids with a $192\,\mathrm{kHz}$ sampling rate and frequencies of approximately 15 kHz, 16 kHz, 17.1 kHz, 18.3 kHz, and 19.6 kHz. The frequencies were chosen to align with Fast Fourier Transform (FFT) bin locations and were spaced unequally to avoid intermodulation components. The synthesized audio signals are converted to analog voltage signals using an 8-channel USB audio interface. The five voltage signals from the audio interface are amplified with independent class D amplifiers to increase the current through the field coils.

Because the field coils present an inductive load to the system, a capacitive tuning adapter is used to remove the imaginary component of the impedance. The tuning adapter consists of five channels, each with a step-up transformer and a bank of parallel capacitors. By tuning the capacitance of each channel while monitoring the voltage and current through the coil, it is possible to maximize the power transfer to the coils.

Prior to measuring the signals from the scleral coil, we amplify them using an instrumentation amplifier (INA128).We then sample the primary and torsional signals using a 4-channel differential input USB oscilloscope from Pico Technology.

Software running in MATLAB on a desktop computer interfaces with the USB oscilloscope and records data at $1\,\mathrm{MSa/s}$. The software buffers the signal for each eye from the primary and torsional coil into 16 ms buffers with 25% overlap. A 4-term Blackman-Harris window is applied to each buffer, since their lengths are not an integer number of periods of the field frequencies. This window function was selected because of its side-lobe attenuation. We then compute the FFT for each window and select the five primary and five torsional components corresponding to the frequency bins of interest. By using only these narrow frequency bins, we also avoid any magnetic interference from the environment or the electronics of the

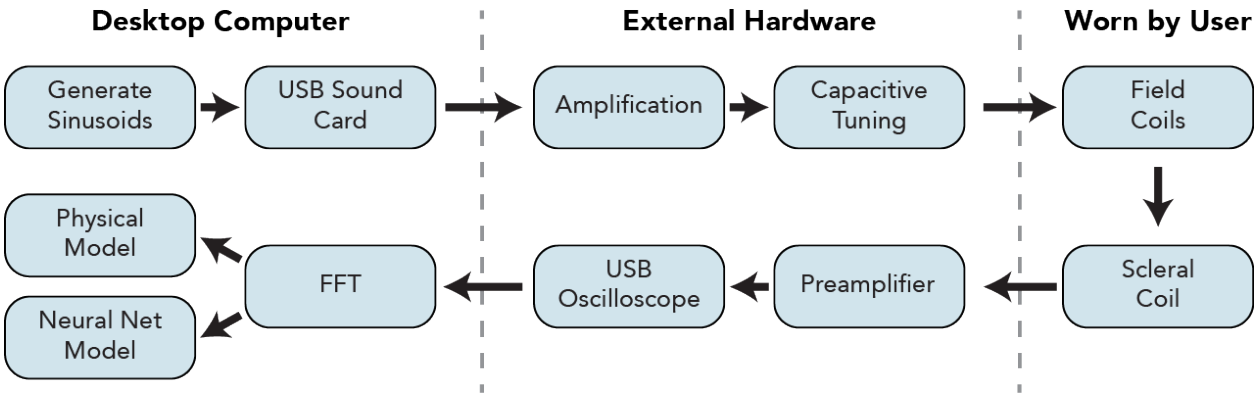

Figure 3.5: Scleral coil tracking signal pipeline. Sinusoids are synthesized from a desktop computer and amplified before passing through the generator coils. This induces a voltage in the scleral coil, which is amplified and processed on a desktop computer.

HMD; the scleral coil measurements are not changed when the HMD is powered on or an application is in use. For each frequency component, we save the magnitude and phase of the complex FFT. Optionally, sequential measurements can be averaged to further improve the signal-to-noise ratio.

In selecting frequencies that align with the FFT samples, we also force each window to contain an integer number of periods of that particular frequency, maintaining the phase of the FFT between windows. However, due to small timing differences between the components of the system, in practice, the phase drifts significantly over time. To account for this drift, we use a 10 second calibration period during which we measure and track the phase drift rate. Periodically, we reset any additional drift that has accumulated. With this technique, the adjusted phase shifts only occur in $180°$ increments when magnetic flux through the coil is reversed due to a change in gaze orientation. We can then reconstruct the signed magnitude by comparing the adjusted phase with the phase at a known gaze orientation.

3.3 Mechanical Test Rig

To evaluate the performance of our scleral coil tracker, we constructed a mechanical 5-DOF test rig, capable of adjusting the orientation of the coil and the position of the HMD with respect to the coil as shown in Figure 3.4 (center, right). The scleral coil is placed inside a plastic holder mounted on a U-shaped arm. Motorized rotation stages from Physik Instrumente allow the arm to swing around (θ) and to tip up and down (ϕ). This allows us to adjust the yaw (θ) and pitch (ϕ) of the coil direction vector with a resolution of $0.0018°$. The HMD and generator coils are mounted on a three-axis linear stage system (x, y, z). This allows us to move the HMD in space with respect to the coils with a resolution of $0.5\,\mu m$, simulating different placements of the HMD on the forehead or slippage of the HMD.

Because we adopt a spherical model of human eye, a scleral contact placed on the surface of the eye moves in space as the eye rotates, according to the radius of the eye (R). For a system with a non-uniform magnetic field, this is an important consideration, as the magnetic field environment will vary with respect to gaze,

even if the HMD is fixed to the user's head. To account for this, the base of the scleral contact is mounted $R_e = 8.5\,\text{mm}$ in front of the center of rotation, simulating the behavior of a human eye. To compute the position offset of the scleral coil, we consider both the offset due to the HMD position (x, y, z) and due to the orientation of the eye (θ, ϕ) according to Equation 3.1.

$$P = \langle x - R_e \sin(\theta)\cos(\phi), y - R_e \cos(\theta)\cos(\phi), z - R_e \sin(\phi)\rangle \tag{3.1}$$

The five motorized stages can be controlled from a desktop computer. We have integrated control of the stages into our MATLAB data collection software to enable an automated sweep of the entire visual field. When measuring the fields, we sweep the coil in the desired range in yaw (θ) and pitch (ϕ) and sweep the HMD position in space (x, y, z). The MATLAB recording software automatically records the scleral coil values one second after stage movement has ceased, to avoid any effects from vibration of the test rig.

Data is collected from $-30°$ to $30°$ in $3°$ increments along the horizontal (θ) and vertical (ϕ) axes. This is comparable to the operating range of other eye tracking devices. We simulate slippage by sweeping the HMD from $-7.5\,\text{mm}$ to $7.5\,\text{mm}$ in $3\,\text{mm}$ increments along the x-axis (side to side), $-5\,\text{mm}$ to $0\,\text{mm}$ in $5\,\text{mm}$ increments along the y-axis (front to back), and $-5\,\text{mm}$ to $5\,\text{mm}$ in $2\,\text{mm}$ increments along the z-axis (up and down), as shown in Figure 3.4 (center). This represents a full sweep of 441 points at $6 \times 2 \times 6 = 72$ locations in space for a total of $31\,752$ data points. Collecting this data takes approximately $8\,\text{h}$.

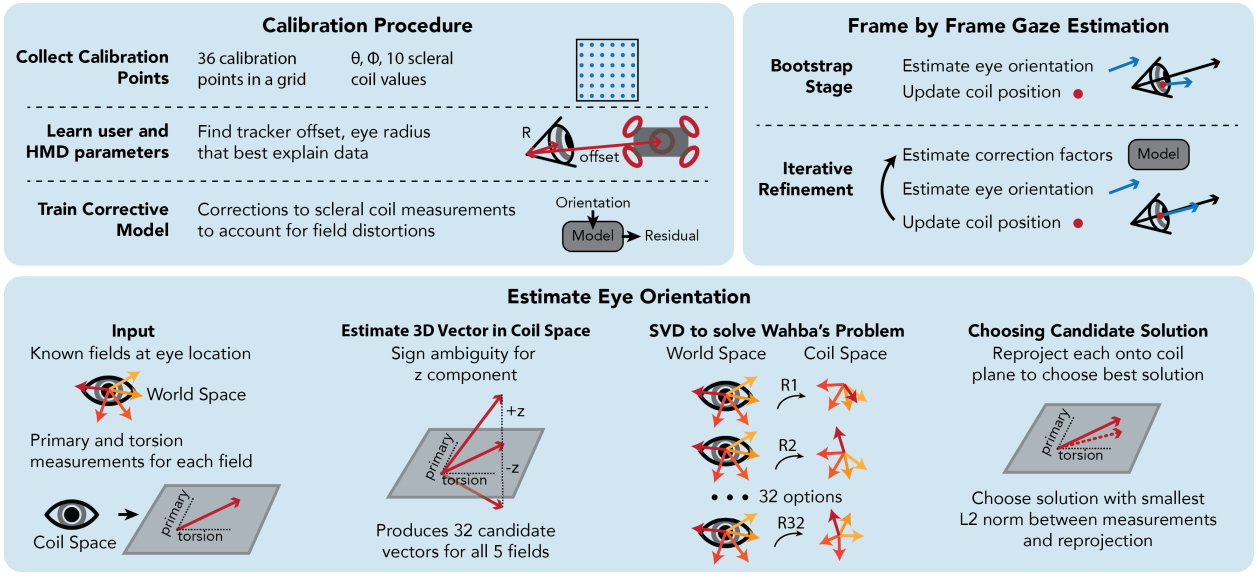

Figure 3.6: Summary of the calibration and gaze estimation procedures. A 36-point calibration procedure is performed when the user puts on the HMD to estimate the HMD offset and eye radius and to train a corrective model for the scleral measurements. Each frame, eye orientation is iteratively estimated and refined using updated coil positions and correction factors. Eye orientation is estimated using an SVD solution to Wahba's problem.

3.4 Gaze Estimation Models

We compare two models for gaze estimation. First, we use a physics approach that models the magnetic fields around the tracker. By comparing the scleral coil measurements with the values we would expect to measure given the magnetic fields, we can estimate the scleral coil position and orientation. We compare this approach to a neural network model that directly estimates orientation given the scleral coil measurements.

3.4.1 Gaze Estimation using Physical Model

The physical model estimates the five magnetic field directions at the eye in world space and then uses the scleral coil measurements to reconstruct an estimate of those vectors in coil space. The eye orientation is determined by the rotation that best accounts for this coordinate system transform. As summarized in Figure 3.6, the model consists of a per-use calibration to estimate the offset of the tracker with respect to the eye and a per-frame gaze orientation estimation procedure.

The five magnetic fields are modeled as wire loops at fixed positions and orientations. Based on the known geometry of the tracker and the current through each generator coil, we can compute the expected magnetic field, $B(P)$, at any potential eye location, according to standard equations for the magnetic field of a coil off the symmetric axis. With this information, given the position and orientation of the scleral coil, we can predict the values we would expect to measure from the scleral coil ($\hat{M}(B, P, \theta, \phi)$).

Calibration Procedure

When the user first puts on the HMD, there is an unknown offset between the tracker and the eye's center of rotation due to differences in user anatomy and HMD placement. Our calibration procedure seeks to estimate this HMD offset. During the procedure, we must also learn the sensitivity of the primary and torsional coils as well as the radius of the eye, as this determines how much the scleral coil moves as the eye rotates. We use a 36-point calibration procedure in which the user looks at targets placed at specific locations on screen (see Figure 3.6, top-left). We simulate this in the mechanical test rig by moving the stages to the required orientations.

Given these calibration points (with a known gaze orientation) and the magnetic field models, we use MATLAB's Global Optimization Toolbox to estimate the scleral coil sensitivities, the offset of the HMD (x_e, y_e, z_e), and the eye radius (R_e). We evaluated our ability to estimate the HMD offset by using the test rig to shift the HMD and calibrating at 72 different offsets within the slippage volume. At each point, we used 36 points for the calibration procedure. Mean Euclidean error in estimating the HMD offset was $0.72\,\mathrm{mm}$ ($\sigma = 0.24\,\mathrm{mm}$).

This model reasonably explains the measurements observed in the scleral coil (M_{meas}). However, there is a small error (mean error for primary coil was 1.5%) in these estimations, caused by not accounting for irregularities in eye movement, irregularities in the magnetic fields due to physical construction, and distortions in the field caused by the presence of the HMD. An important realization, however, is that these errors are systematic. It is possible to model the residuals between each of the ten scleral coil readings and

those predicted by the physics model ($M_{meas} - \hat{M}$) as a function of gaze location. That is, if we knew the user's gaze, we could adjust the measurements from the scleral coil such that they would best fit the predicted values from the physics model. When modeling the residuals as a function of gaze position, it is possible to learn the corrective model using just the 36 calibration points. We use MATLAB to fit a function ($F(g_x, g_z) = \hat{M} - M_{meas}$) to the 36 calibration points using biharmonic interpolation, for each of the 10 scleral coil measurements. During gaze estimation, we rely on an iterative bootstrapping approach to refine the gaze estimate using the corrective models.

Frame by Frame Gaze Estimation

When estimating the orientation of the eye, we first use the magnetic field model to compute the five expected magnetic field directions at the scleral coil position, which can be calculated using the HMD offset learned during the calibration. Initially, since we do not know the eye orientation, we do not account for the traversal of the coil in space as the eye moves; we assume, for now, a fixed position of the coil. Next we reconstruct the five measured field directions in coil space from the scleral coil measurements. We consider the scleral coil as biaxial sensor that outputs the projection of the magnetic field vector in the plane of the scleral coil, which is determined by the orientation of the primary and torsion coils (see Figure 3.6, bottom). To reconstruct the third component, we normalize the measurements by the field strengths and by the relative sensitivities of the primary and torsional coils, so that the scleral coil sensor effectively reports two components of a unit vector. We can reconstruct this third component as $M_z = \pm\sqrt{1 - M_p^2 - M_t^2}$. Note that there is a sign ambiguity for this third field component. We consider both possibilities for each of the five fields, producing 32 candidate sets of field measurements.

For each candidate set, we now have five known field vectors (from the physics model) and five estimates of those fields (from the scleral coil, with the 3rd component estimated) in the unknown reference frame of the coil. This is an instance of Wahba's problem, which seeks to find a rotation matrix between two coordinate systems given a set of weighted observations. We use an SVD solution to Wahba's problem to estimate the gaze [94]. The weights for each field are set based on a precomputed average signal-to-noise ratio for each coil.

This procedure outputs 32 candidate solutions that fully specify the eye orientation. To choose among the solutions, we compute the expected values of the scleral coil measurements for each of the candidate solutions and choose the solution that most closely matches the measured values.

This initial gaze estimation does not account for the corrective model or the movement of the eye as it rotates. Given the initial gaze estimate, which tends to be 3° off on average, we can compute an estimate for the eye position and necessary corrective factors. To refine our gaze estimation, we compute the updated magnetic field directions at the new eye location, adjust each of our scleral coil measurements according to the corrective model ($M_{adj} = M_{meas} + F(\hat{g_x}, \hat{g_z})$) and recompute the estimated gaze orientation. This next gaze estimate is much more accurate and allows an even better estimate of the corrective terms and scleral coil position. We repeat this process until the estimated gaze converges; typically this requires 5 iterations. Applying this procedure to the 405 test orientations across all 72 HMD offsets, we achieve a mean error of $0.18°$. The model performance for a small subset of these points is depicted in Figure 3.7 (top right) and a complete summary is shown in Figure 3.8 as a cumulative distribution function (CDF).

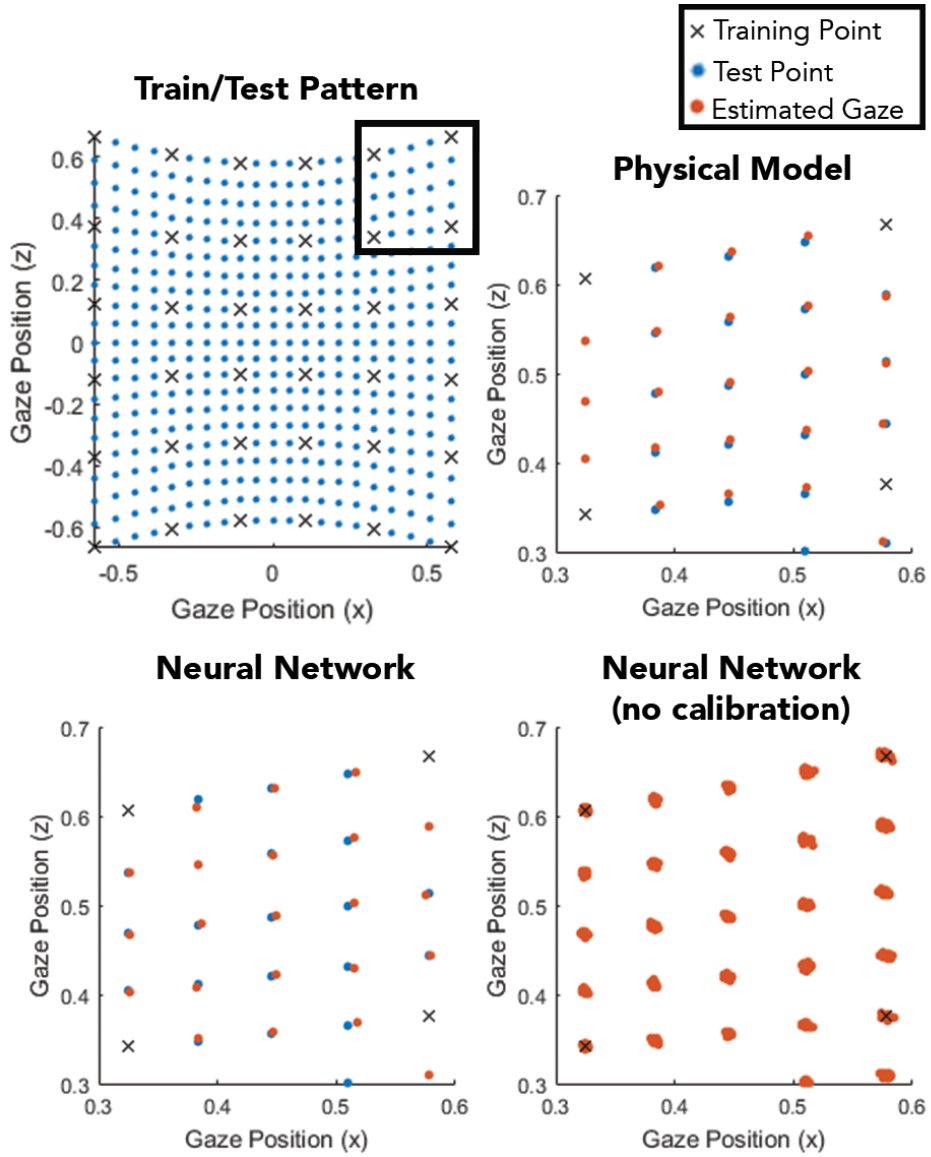

Figure 3.7: Gaze plot showing training (black X), test (blue circles), and estimated (orange circles) gaze locations as projected onto a plane at $y = 1$. (top left) 36 points were used for training and the remaining 405 used for testing. (bottom and right) The physical and neural network model gaze estimations and ground truth for a small subset of the field of view. (bottom-right) The calibration-free neural network shows gaze position for all 72 HMD offsets.

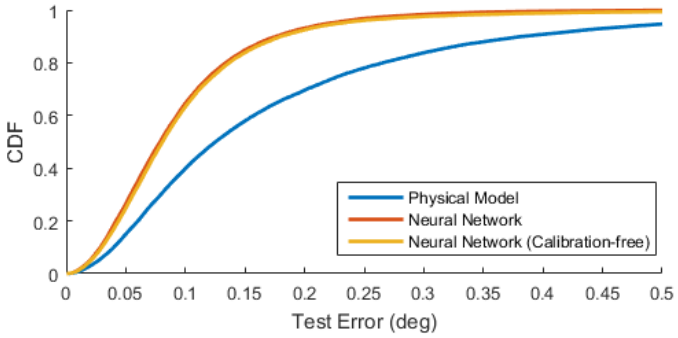

Figure 3.8: CDF of gaze orientation accuracy across all positions with the physical model, neural network model, and calibration-free neural network model. The physical model achieves $0.18°$ mean error. The neural network models both outperform the physical model with mean errors of $0.094°$ and $0.099°$, respectively.

3.4.2 Neural Network Model

Rigid Body Model

As an alternative to the physical model, we trained a neural network to learn the effects of field divergence, distortion caused by the HMD, and movement of the eye. To train the model, the user performs a short 36-point calibration procedure. Each data point consists of the 10 scleral coil measurements. Forty neurons were used for the hidden layer and the output layer consists of two neurons, which were trained to output the orientation of the scleral coil (θ and ϕ).

As with the physical model, we used the test rig to evaluate the accuracy of this gaze estimation model. At each of the 72 HMD offsets within the slippage volume, we train a separate network using 36 training point and holding out the remaining 405 points for testing. This approach achieves $0.094°$ mean error, averaged over all HMD offsets and test points. Gaze estimations for one HMD offset are shown in Figure 3.7 (bottom left). The full results are summarized as a CDF in Figure 3.8.

Calibration Free Model

We also sought to explore whether the neural network can be trained in a position-independent manner in order to reduce sensitivity to slippage. In this approach, a two-stage model is used to both estimate the search coil position and orientation. A significant amount of training data is needed to train these models, making a calibration procedure unfeasible. Moreover, it is impossible to position the search coil at a specific location in space while it is being worn by a user. Hence, we introduce a calibration-free tracking technique using a pre-trained model.

First, a neural network with 10 input nodes (corresponding to the 5 primary and 5 torsional observations), 40 hidden nodes, and 3 output nodes (x, y, z) is trained using calibration points collected at 32 of the 72 HMD offset positions. Mean Euclidean error on the remaining $27\,880$ points was $0.084\,\mathrm{mm}$.

We now augment our original gaze model with the position estimation. Input consists of 13 nodes, 10 from

the scleral coil and the estimated x, y, z location in space. Again, we train on calibration points collected the training positions and test on the remaining points. Mean error on the test set was $0.099°$. These results are summarized in the target plots in Figure 3.7 (bottom right) and in the CDF in Figure 3.8.

3.5 Discussion

We present two gaze estimation methods based on a physics model and a neural network. The physics model achieves a mean gaze orientation error of $0.18°$. By modeling the magnetic fields, this model allows us to understand the magnetic environment of the scleral coil and lends itself well to extensions. For example, future work could look at online calibration procedures, torsion estimation, or using the data from both eyes in calibration. These applications are more difficult with the more opaque neural network model.

With a mean error of $0.094°$, the neural network model is better able to approximate the magnetic environment around the tracker and the distortions caused by the HMD. The calibration-free neural network model can significantly improve usability by estimating both the HMD slippage and eye orientation in each frame. However, since this model was trained on data collected from the test rig, it is unclear how well it generalizes to a human user in a different magnetic environment. Experiments are underway to validate the performance of this technique with human participants.

A promising avenue for future exploration is the combination of these models. By providing a neural network with the magnetic field estimations from the physical model, we can simplify the problem and enable shorter training times.

In the calibration-based models, we expect the user to perform the 36-point calibration procedure when they first put on the HMD and whenever the HMD shifts on their face. We have implemented a shift detection algorithm that can automatically prompt the user to recalibrate, if needed.

Our system samples the scleral coil signal at $1\,\mathrm{MSa/s}$. After buffering and windowing, the gaze estimation is output at a rate of $244\,\mathrm{Hz}$. Significant increases in output data rate can be obtained by reducing the window size or increasing overlap width. We intend to implement a native code version of the algorithm or use special purpose programmable logic hardware. Similarly, we can increase temporal resolution by using higher frequency sinusoids and taking advantage of additional bandwidth. We also explored temporally averaging samples together, as is common in many SSC tracking systems. Though we can improve the accuracy on the neural network models to $0.03°$, we chose maintain a higher data rate instead.

In our evaluation, we focused on the mechanical test rig as a proxy for a human user. This allows us to collect a comprehensive dataset of positions and orientations and gives us reliable ground truth references that can isolate the accuracy of the tracker, independent of the fixation accuracy of a human user. In designing the test rig, we accounted for many nuances that make eye tracking in an HMD difficult. For example, we accounted for movement of the scleral coil in space by offsetting the scleral coil location from the center of rotation of the test rig. We also accounted for HMD slippage by mounting the HMD and tracker on a 3-axis translational stage system. Since for many VR applications, measuring the gaze position is sufficient, we did not prioritize evaluating eye torsion, which would provide a full-attitude gaze estimation. Though our physical model provides an estimate of torsion, we do not evaluate its accuracy because the test rig only allows us to adjust the torsion of the scleral coil in $10°$ increments. A future version

of this test rig could include precise control of torsion in the -4° to 4° range, which would enable a more rigorous evaluation of the torsion estimation from our model. However, the calibration-based models do account for systematic torsion that occurs regularly as a function of gaze.

3.6 Conclusion

EyeContact is a scleral coil tracking system designed for a virtual or augmented reality HMD that enables high-speed, high-accuracy mobile eye tracking without instrumentation of the environment. Our mechanical test rig enables calibration and evaluation with a reliable ground truth. We describe a calibration procedure and two different approaches to estimate gaze: a physical model with 0.18° mean error and a neural network model with 0.094° mean error. We hope this system will be a useful tool for researchers in need of high-quality eye tracking within an HMD.

4 LOW-POWER EYE TRACKING

4.1 Introduction

While EyeContact prioritized accuracy and precision, low-power gaze tracking is an important part of the design space of eye tracking devices, particularly for standalone HMD devices where power and compute is at a premium. This work explores the use of multiple low-power photosensors to estimate gaze.

Most eye tracking systems rely on video-oculography, the use of cameras to capture images of the eye many times per second. An imaging sensor uses a dense array of photosensors behind a focusing lens and produces millions of pixels per image. However, most of these pixels are completely irrelevant to estimating gaze. Imaging sensors consume significant power and rely on extensive computation, often using convolutional neural networks [71], to track the eye. Instead, this project relies on individual photosensors placed around the eye and infrared illumination that reflects off of the eye. Because the surface of the cornea is non-spherical and the pupil, iris, and sclera have different infrared reflectance properties, the amount of light that is received at a particular photosensor varies with gaze.

As a simplified example, consider a narrow field of view photodiode directed at the eye and assume uniform infrared illumination. When the pupil, which appears dark in infrared, falls within the field of view of the photosensor, the observed signal will be low. Conversely, when the sclera, which appears bright in infrared, falls within the photodiode field of view, the observed signal will be high. In practice, the output signal is proportional to the ratios of iris, pupil, and sclera in the field of view. As the eye rotates, these ratios change, creating a signal that can be used to reconstruct the eye's rotation.

To better understand this problem, we created a simulation environment that consists of a scanned 3D head model and realistic, parametric, 3D human eye model. We developed a pipeline to simulate the measurements from any photosensor-based tracking device. We use the simulation technique to prototype our gaze estimation algorithms and to optimize the position and orientation of the photosensors.

Our first prototype device consists of eight photodiodes spaced around the eye along with three infrared LEDs for illumination. An infrared hot mirror in front of the eye allows us to place a camera outside of the field of view that can view the eyes for debugging and diagnostic purposes. We use a photometric front-end chip to drive the LEDs and photodiodes. By modulating the LEDs at a particular frequency and using synchronous detection on the photodiodes, we maximize sensitivity and reduce the effects of variable environmental illumination. We use a five layer neural network to map sensor values to gaze directions. We present a second device that incorporates the design into a head-mounted display and adds additional sensors to measure slippage of the headset on the face.

Power and computation represent the most significant advantages of our technique. The analog portions of our device (excluding IO) consume $16\,\mathrm{mW}$ while operating at $400\,\mathrm{Hz}$, with the potential to increase or decrease power depending on speed and accuracy needs As a comparison, the open source Pupil Labs eye tracking system consumes $900\,\mathrm{mW}$ per eye.

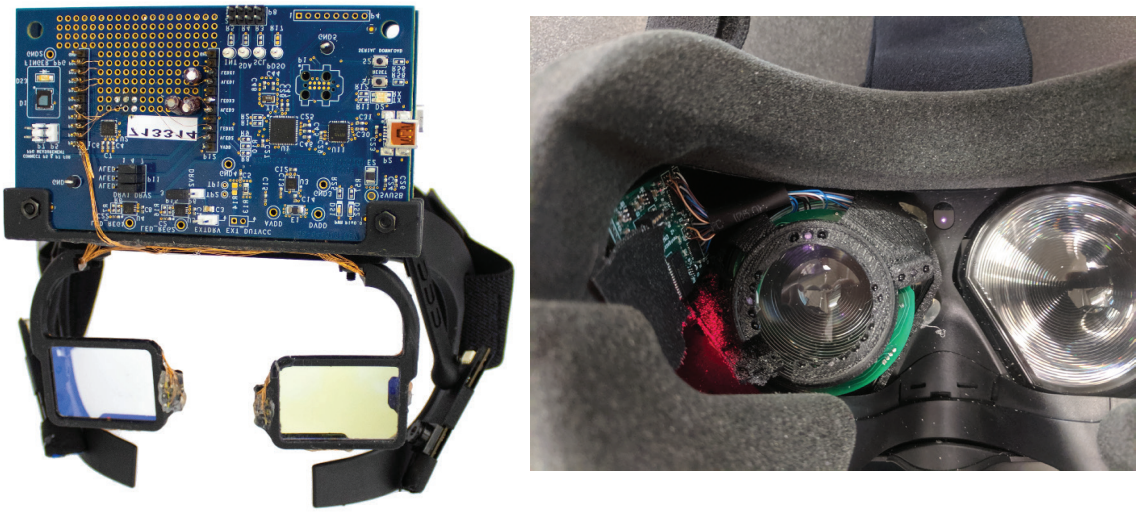

Figure 4.1: Two prototype photosensor-based eye trackers. (left) a face-mask binocular tracker with 8 photosensors per eye. (right) a fully-integrated monocular HMD eye tracker with twelve eye-tracking sensors and four sensors to monitor slippage on the face.

Our contributions in this work include:

1. a rendering-based framework for simulating arbitrary photosensor-based eye tracking devices with new insights into the nonlinear behavior of eye reflections,
2. a prototype implementation and evaluation of an 8-sensor mask-based eye tracking system,
3. and a prototype implementation of a second 16-sensor HMD-integrated eye tracker that also tracks slippage of the device on the face.

4.2 Background

4.2.1 Principles of Photosensor-based Tracking

Camera-based eye tracking and photosensor-based eye tracking both rely on infrared (IR) illumination and an IR-sensitive sensor. Under infrared illumination, the pupil appears dark, while the iris, sclera, and skin appear bright. Figure 4.2 shows an eye illuminated by two infrared LEDs and captured by an IR-sensitive camera. An imaging sensor produces a high-dimensional signal where light's angle of arrival is encoded spatially as pixels. A single imaging sensor will often produce millions of pixels, which can be used to track pupils or glints on the eye. Capturing and processing data requires significant power and computation.

In contrast, a photosensor, such as a photodiode or phototransistor, consists of a single light-sensitive semiconductor element behind a lens. This produces a one-dimensional signal that aggregates light from an arrival cone. By adjusting the lens in front of the semiconductor element, one can control the size of the

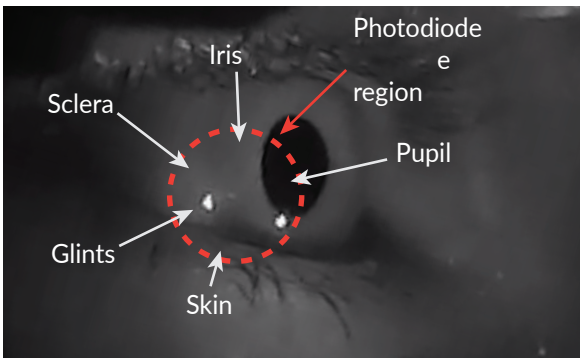

Figure 4.2: An image of the eye illuminated by two infrared LEDs and captured by an IR-sensitive camera. The highlighted circular region indicates a possible sensitivity region for a photosensor placed in front of the eye. The position of glints and the size of the pupil, iris, sclera, and skin within the region will determine the photosensor output.

sensitivity cone. This region often exhibits a Gaussian shape and is expressed as an angle of half-sensitivity. The circular region in Figure 4.2 shows a possible sensitive region for a photosensor placed in front of the eye. Depending on the orientation of the eye, the photosensor will capture light directly reflected from cornea and scattered from the iris, sclera, and skin. Light hitting the pupil will be mostly absorbed.

The primary consequence of this, is that as the pupil moves to fill the sensitive region, the amount of light measured by the photosensor will decrease. Because the glints are so bright compared to scattering effects from the rest of the eye, a secondary effect arises that corresponds to the position of the glint(s) within the Gaussian sensitive region. Because the eye is not perfectly spherical, the glint position and orientation changes with the orientation of the eye. If the eye is illuminated by a single LED, the strength of the signal corresponding to the glint reflection is a function of how close the path lies to both the optical axis of the LED and photosensor. A direct, on-axis reflection straight into the photosensor will exhibit the strongest response, due to the Gaussian shape of the emission and sensitivity cones.

Because the eye can move horizontally and vertically, at least two distinct sensors are required to reconstruct gaze. However, for robustness, photosensor-based tracking approaches generally use four or more sensors. In this work, we present one prototype using eight photodiodes and a second version using sixteen.

4.2.2 *Simulation Framework*

To better understand this behavior and enable experimentation, we first constructed a framework to simulate photosensor-based tracking. We started with the rendering-based gaze estimation framework designed by Kim et al [71]. This framework uses a 3D scan of a human face with a parametrically-defined eyeball. The eyeball can be rotated to any horizontal and vertical gaze direction and the pupil size can be adjusted from $2\,\mathrm{mm}$ to $8\,\mathrm{mm}$. Moreover, the top and bottom eyelids can be adjusted from fully open to fully closed. The textures have been adjusted to match properties of the skin under infrared illumination.

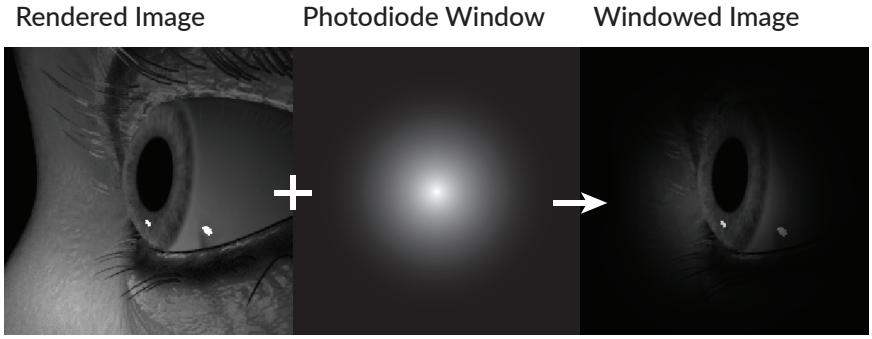

Figure 4.3: The simulation pipeline takes a rendered image from the point of view of a photosensor, applies a windowing function to simulate the photosensor lens, and sums the resulting image to simulate sensor output. This image has been artificially brightened to show detail, but would actually be rendered much darker at 16-bit resolution to prevent clipping.

To simulate photosensor-based tracking, we use the Blender 3D environment to place cameras at the position and orientation of the photosensors we wish to simulate. We model the infrared LEDs as white point light sources. For a given facial configuration (eye gaze in x and y, pupil size, eyelid position), we render an image from the viewpoint of each of the cameras. An example image from a single camera is shown in Figure 4.3. For each image, we transform all pixels in the image using the following Gaussian window function:

$$g(x,y) = e^{\frac{-\left((x-x_0)^2+(y-y_0)^2\right)}{2\sigma^2}} \tag{4.1}$$

where x and y represent the pixel coordinates, x_0 and y_0 represent the image centers, and σ is proportional to the photosensor width, in pixels. Figure 4.3, right, shows the effect of this transform. Finally, to simulate the photosensor output, all pixels in the transformed image are summed as follows, where $i(x,y)$ represents a pixel from the original rendered image:

$$s = \sum_x \sum_y i(x,y)g(x,y) \tag{4.2}$$

An important observation from this process was the importance of using 16-bit rendering and taking care not to oversaturate the image. The images in Figure 4.3 have been artificially brightened, for clarity, but note that the glints in these brightened images consist of saturated white pixels. If these images were used for the simulation, the signal due to the glint would be artificially weakened. To achieve a high-fidelity simulation, the simulated illumination must be decreased so that there are no clipped pixels from the direct reflections. Similar simulation techniques that either use rendered images[193, 135, 136] or images captured from a camera [68] could be prone to this issue.

A second observation concerns the interaction of the glints with the edge of the cornea as the eye changes concavity. As the eye moves and a glint approaches the cornea, the glint becomes stretched and the received signal strength at the sensor increases. The eye orientation at which this effect occurs depends on the

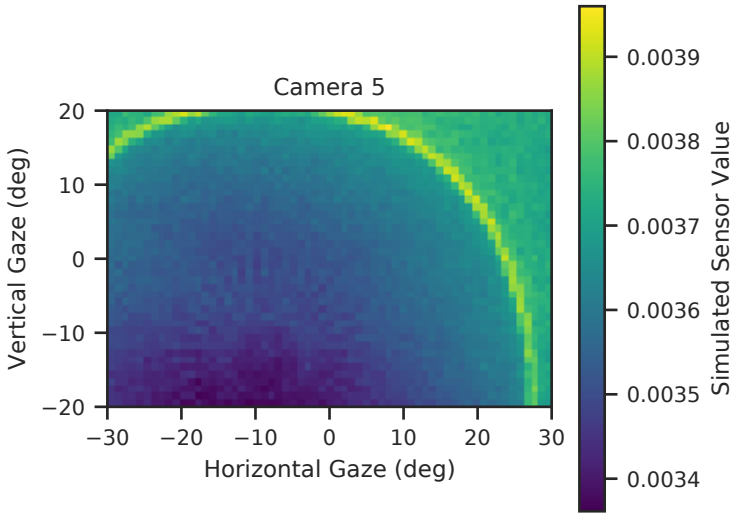

Figure 4.4: The simulated output of a single photosensor as a function of gaze. Brighter colors indicate a higher response. The bright circular outline around the periphery is uniquely captured by this simulation and represents a direct reflection from the edge of the cornea.

position of both the LED and photosensor. This effect is illustrated more clearly in Figure 4.4. This figure shows the output of a single simulated photodiode as a function of gaze direction. The relatively smooth gradient along the lower left portion of the image is due to the pupil and glint moving within the sensitive region. The circular edge along the outsides of the image corresponds to gaze locations where the glint is positioned at the edge of the corneal bump. Within this bright edge, the glint is positioned on the cornea; outside of this edge, the glint is located on the sclera.

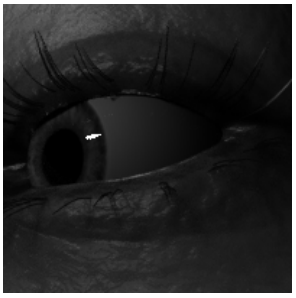

Figure 4.5: The simulated source image corresponding to a smeared corneal reflection. The gaze coordinate is 27° horizontal, −20° vertical.

Figure 4.5 shows an artificially brightened rendering for the gaze location of 27° horizontal and −20° vertical. The smearing of the glint along the edge of the cornea causes more light to enter the photosensor.

Compare these simulated results to actual collected data as the eye pursues a target along a horizontal line from $-20°$ to $20°$ as shown in Figure 4.6. The spike around $10°$ corresponds to the glint aligning with the edge of the cornea. This effect has not been demonstrated or accounted for in prior work that relies on simulated data [193, 135, 136, 68].

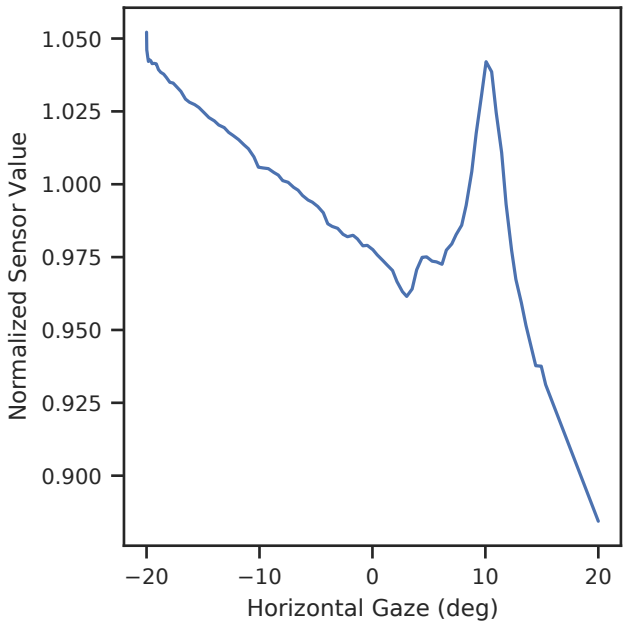

Figure 4.6: Actual data from a photodiode during a horizontal sweep. As the simulation predicted, there is a distinct spike as the gaze moves off center.

A practical implication of this result is that it is best to minimize the number of point light sources active at a time. Multiple LEDs will result in multiple glints that only complicate the tracking problem. Fewer LEDs or diffuse illumination would maximize the smoothness of the transfer function. On the other hand, future work could consider using these direct reflections as a feature to model the geometry of the eye.

4.3 Hardware

4.3.1 System Architecture

Our near-eye photosensor-based eye tracking system consists of a ring of LEDs and photodiodes around each eye. Though the system is intended for use with a near-eye display, for prototyping and debugging purposes the device was constructed as part of a separate face mask. The mask attaches to the face using an elastic band. An external display is used to calibrate the gaze tracker.

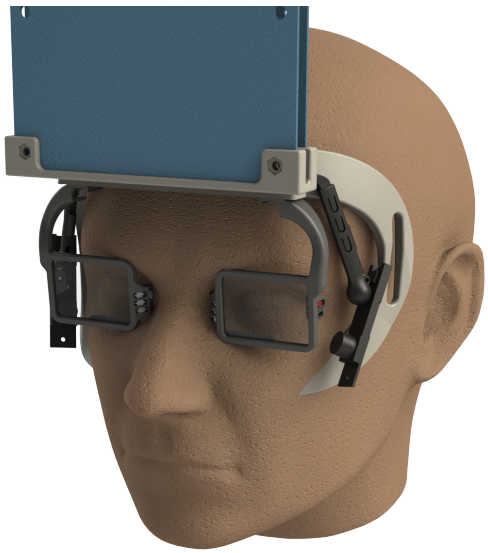

Figure 4.7: A rendering of the binocular face mask-based eye tracking device. Each eye frame contains eight photodiodes to track eye movement. Cameras mounted to the side of the eye capture eye images through the hot-mirror. Signal processing electronics are placed above the eyes.

The full design is shown in Figure 4.7 and Figure 4.8. The eye is illuminated by three LEDs embedded within the frame of the device. Two or more photodiodes are clustered around each LED. The eight photodiodes are strategically placed such that for a typical user, they will cover different parts of the eye. Figure 4.9 shows the results of a Zemax optical simulation for a subset of six photodiodes. The images highlight the intended sensitive region for each photodiode.

To facilitate development and proper placement of the optical elements, a Pupil Labs infrared camera is placed just outside the users field of view. A hot mirror in front of the eyes reflects infrared light from the eyes into the camera lens. Simultaneous capture of the camera and photosensor signals enables a better understanding of trends and artifacts in the photosensor signals. Though this causes some distortion of the user's view, it is minor and can be removed if the camera is not needed.

The LEDs emit and the photodiodes are sensitive to $940\ \mathrm{nm}$ infrared light. This particular wavelength is less prominent in outdoor lighting [11] due to absorption by water molecules in the atmosphere. Nonetheless, it is still important to use modulated illumination to reject any ambient light. The device uses an analog front end (ADPD103) to synchronously modulate the LEDs and sample the photodiodes. Each LED is illuminated for $3\ \mu\mathrm{s}$ every $24\ \mu\mathrm{s}$. The photodiode response is bandpass filtered and synchronously integrated. The result is a photodiode signal sensitive to changes in reflected light from the LEDs, but not from other light sources.

To minimize the effect of direct reflections off the cornea, only one LED is illuminated at a time. In this implementation, only two LEDs are used in a particular session; each LED is associated with the four nearest photodiodes. For a single frame of data, the first LED pulses four times while the four nearby photodiodes are integrated and sampled. This process repeats for the second LED and remaining photodiodes. The

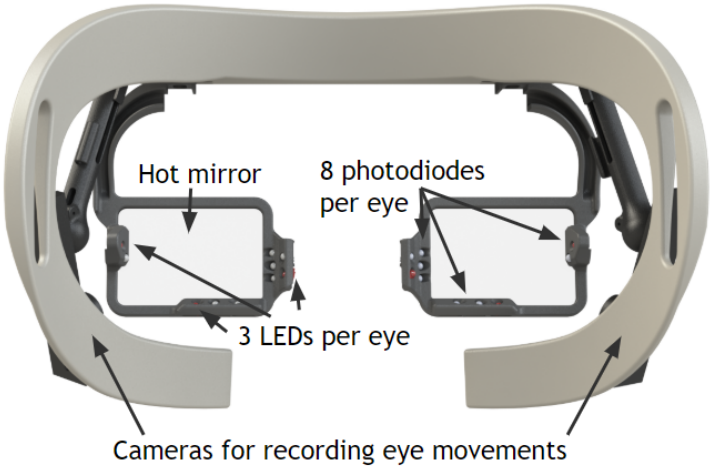

Figure 4.8: Layout of the LEDs, photodiodes, and cameras, as seen by a user.

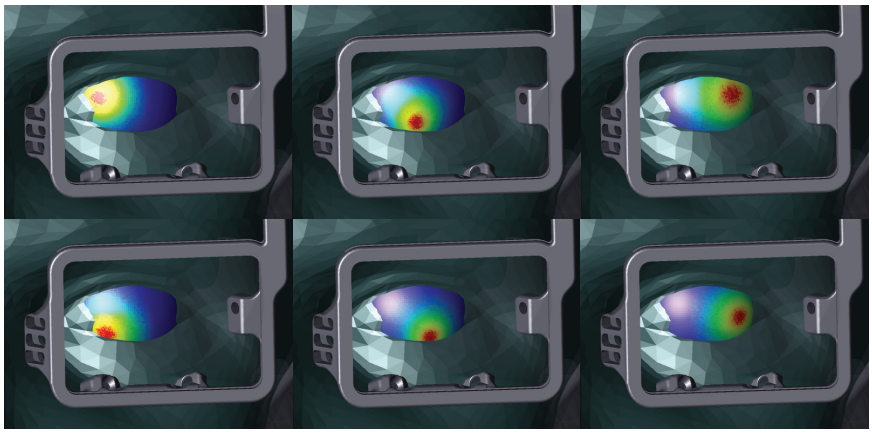

Figure 4.9: A subset of six photodiodes and their projected sensitive regions on the eye as simulated by Zemax.

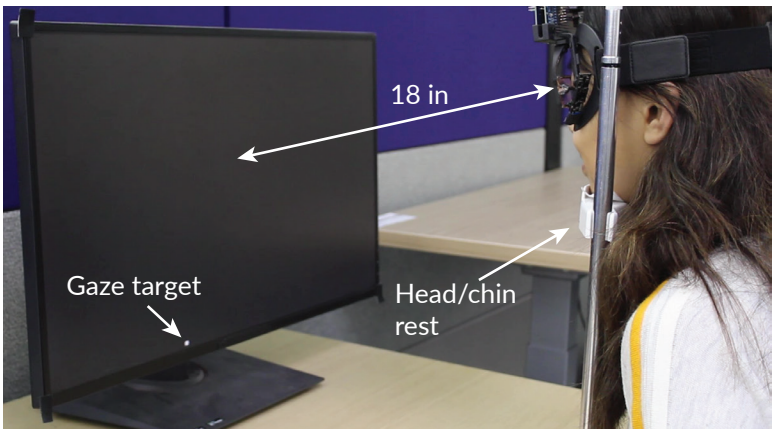

Figure 4.10: The calibration procedure requires a user to rest their chin in a head rest 18 inches from a fixed display, upon which a gaze target is shown.

overall data rate is determined by the number of LED pulses and delay between LEDs. In this prototype, the output data rate was configured to be $400\,\mathrm{Hz}$. The LED current is also configurable and determines the signal to noise ratio. In this prototype, it was set at $67\,\mathrm{mA}$. Note that this is instantaneous current through the LEDs. At this current setting, the overall power consumption of the AFE and LEDs is only $16\,\mathrm{mW}$.

The analog front end is placed on development boards mounted above the eyes on the facemask. Each front end supports eight photodiodes, so one is used for each eye. The photodiodoes and LEDs are wired up to the AFEs using thin magnet wire along the edges of the frame (see Figure 4.1).

4.4 Calibration and Modeling

4.4.1 Calibration Procedure

Because the device does not have an integrated display, it is calibrated using an external monitor. The calibration setup is shown in Figure 4.10. While wearing the device, a user places their head in a desk-mounted chin rest. A monitor is placed $18\,\mathrm{in}$ from the face. While such a setup may be insufficient for between-session tracking, it keeps the head fixed enough for within-session tracking.

A circular gaze target with radius $10\,\mathrm{px}$ ($1.2\,\mathrm{mm}$) is placed on the screen for calibration. The x and y coordinates of the target in screen-space can be computed from the screen distance and desired angular gaze coordinates. The target is presented using Vispy and Qt5.

Traditional calibration procedures ask the user to fixate on a number of targets and use this data to construct a mapping onto gaze location. In this system, the mapping function, as shown in Figure 4.4 is highly nonlinear. To maximize the amount of data available to learn this function, we rely on a smooth pursuit calibration task, following the guidelines outlined by Pfeuffer et al [121]. The gaze target travels in a series of linear paths over $\pm 20^\circ$ vertical and horizontal field of view as outlined in Figure 4.11. For each segment,

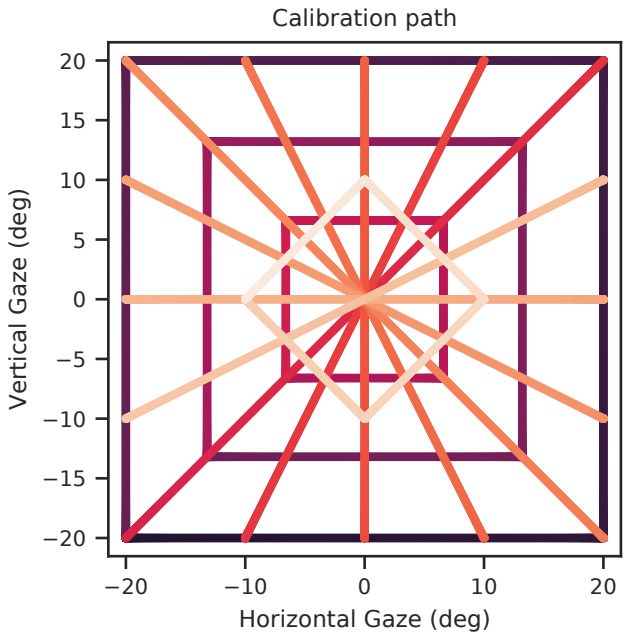

Figure 4.11: The path used for the smooth pursuit calibration. Calibration begins at the bottom left corner and ends at -10° horizontal, 0° vertical.

the target smoothly accelerates from rest over $1\,\mathrm{s}$, up to a max speed of 6° per second and then decelerates over $1\,\mathrm{s}$ back to rest at the end of the segment. This produces a dense sample of the gaze space. Particular care was taken to optimize the display pipeline to minimize any jitter of the target as it moved, as this could cause undesired saccades during the pursuit.

When analyzing the data, a preprocessing step filters out any blinks by looking for instances where there is an abrupt change in the photodiode signals. The remaining signal is downsampled to keep 100 points along each line segment to avoid overfitting points along the line.

4.4.2 *Eye Gaze Model*

The gaze model maps the photodiode output to gaze coordinates, in degrees. Because the mapping is highly nonlinear, we leverage a neural network model to map the eight photodiode signals to 2D gaze. We first scale all photodiode outputs to 0 to 1 and then apply PCA to reproject the data. The transformed signals are input to a network with four hidden layers of 64 nodes each and tanh activation functions. Training is performed using scikit-learn with a batch size of 4.

After calibration, the gaze is estimated and shown on screen in real time. A post-model exponential filter ($\alpha = .2$) is applied to smooth the estimation results.

Task	Mean Error	Standard Deviation
Smooth Pursuit Validation	1.68°	0.56°
Fixation Validation ($\pm 20^\circ$)	2.67°	0.98°
Central Fixation Validation ($\pm 10^\circ$)	2.35°	0.58°

Table 4.1: Performance of the gaze model on different test sets. The model performs best on the smooth pursuit task.

4.5 Evaluation

4.5.1 Procedure

To evaluate the performance of the eye tracking device, we conducted a preliminary study. We invited six participants (4 male, 2 female) to try the device by performing a calibration and evaluation task. After a brief introduction, the researcher helped place the device on the user's head. Then the user carefully placed their chin in the head rest and ensured they could comfortably see the screen.

A target was displayed on-screen and smooth pursuit calibration proceeded per the path in Figure 4.11. Immediately following the calibration, the system displayed 25 targets on a grid within $\pm 20^\circ$ horizontal and vertical. The first twenty segments of the smooth pursuit data were used for training. The last four segments as well as the 25 fixation targets were used for testing. After the data was collected, the neural network model was trained and participants were able to qualitatively judge the quality of the tracking by observing the real time gaze estimate on the screen.

4.5.2 Results

The results from this evaluation are shown in Table 4.1. Mean error on the last four segments of the smooth pursuit task was 1.68°. The error increased to 2.67° on the fixation task, but is slightly better when limiting the data to $\pm 10^\circ$. Among the six trials, the best error on the $\pm 20^\circ$ fixation task was 1.1° and the worst was 4.2°.

Figure 4.12 shows a example validation result from one participant. The blue dots represent the fixation target and the orange dots represent the estimated gaze location.

4.6 Towards an Integrated VR Eye Tracker

Although the face mask prototype was designed with a head-mounted display in mind, there are a number of challenges that can only be explored by fully integrating the tracker into the HMD. Proper placement of the sensors and LEDs becomes critical in order to maximize the field of view. There is also constrained space within the headset to put additional sensing electronics. A fully-integrated solution obviates the need for a chin rest, as the sensors are rigidly coupled to the display. However, there is no guarantee that

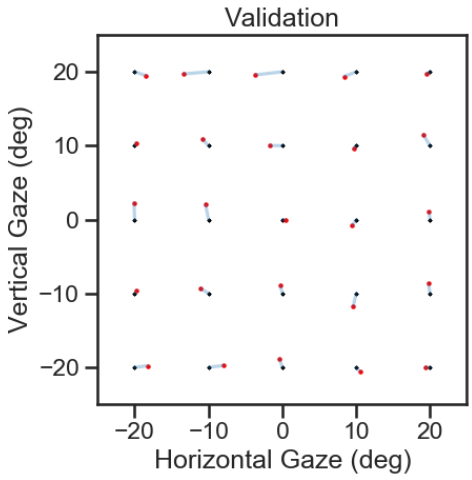

Figure 4.12: An example result from a fixation validation trial that was trained on smooth pursuit data. This example has a mean error of 1.6°.

the display is rigidly coupled to the face, so detecting and accounting for HMD shifts/slippage becomes paramount.

4.6.1 Hardware

To explore these challenges, we constructed another version of the device integrated into an HTC Vive HMD. Informed by both simulation analysis and the face mask prototype, we developed a system that incorporates 12 eye-tracking photodiodes and 4 face-tracking photodiodes. The components of this device are illustrated in Figure 4.13. The eye-tracking photodiodes are positioned mostly below the eye to minimize the effects of the upper eyelid and eye lashes. To monitor and account for slippage, four additional face-tracking diodes and two LEDs are placed on the top of the frame. Two are pointed toward the nose bridge and two are pointed just above the eyeball. These are designed to capture both horizontal and vertical slippage, independent of eye motion.

The optical elements are mounted within a 3D printed frame designed to fit around the lens of the HMD. The frame extends forward and slightly inward to reduce the eye relief and bring the optical elements closer to the eye's optical axis. This unavoidably reduces the field of view of the HMD to approximately $\pm 30°$, depending on the user's head shape. The optical elements connect to a custom circular backplate PCB to facilitate wiring. Twenty-four thin wires connect the backplate PCB to sensor PCB, mounted on the side of the HMD. The sensor board consists of two ADPD1080 AFEs which communicate with an MSP430 via I2C. An on-board FTDI chip allows the board to stream data to a PC over USB. For debugging purposes, the frame also contains a mount for an IR camera that can be used at larger eye reliefs.

Each AFE drives two LEDs in separate time slots. Altogether, four LEDs are time-multiplexed in four separate time slots and each slot samples from four photodiodes simultaneously. The overall data rate is

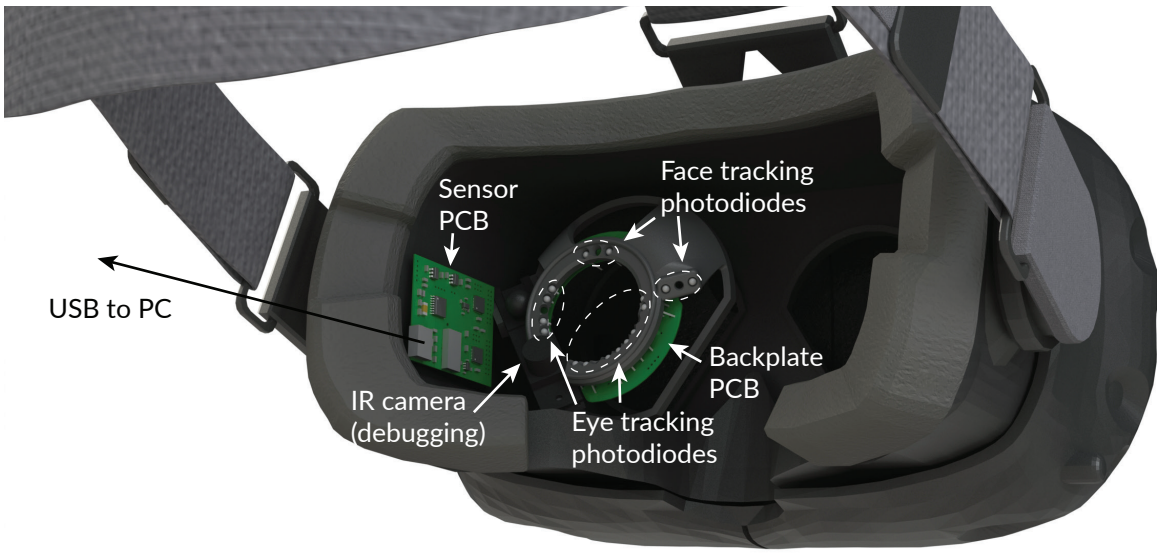

Figure 4.13: A second prototype features an HTC Vive with a fully-integrated eye tracker. Four sensors oriented away from the eyes are used to track the slippage of the device on the face.

customizable, but for prototyping purposes was set at 225 Hz.

4.6.2 Calibration

The calibration procedure for this device resembles the procedure describe in Section 4.4. Notably, no head-rest is required for this calibration since the display is coupled to the eye. A gaze target is displayed on-screen using the Unity engine. To eliminate the effect of head motion, we disable head tracking so that the virtual environment is locked to the user's head. Because this has the potential to cause motion sickness when users move their head, we ask users to rest their chin on their arms to keep the head relatively stable. The gaze target spans 1.2° in a shape defined by the optimal gaze target as described by Thaler et al [161] and shown in Figure 4.14. The 2D gaze target is presented on a 3D dome and is rotated such that it is always directly facing the user.

4.6.3 Preliminary Evaluation

As a proof of concept, one user completed six calibration procedures as described in Section 4.5.1. The fixation task bounds was reduced to $\pm 15^\circ$ since the quality of the display past this point made calibration difficult. Aggregate results from these trials are presented in Table 4.2. The fixation task performance was similar to that measured in the face mask prototype, but the smooth pursuit performance was degraded. However, the system must be evaluated on more users before meaningful comparisons can be drawn.

An example fixation task result is shown in Figure 4.15.

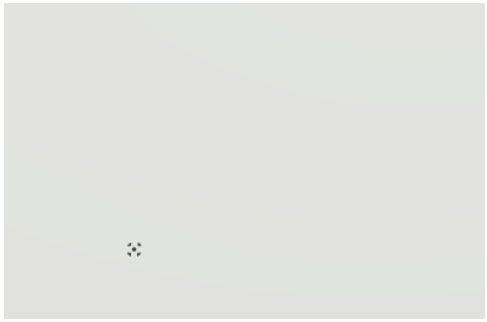

Figure 4.14: The calibration target was displayed in Unity on the HMD.

Task	Mean Error	Standard Deviation
Smooth Pursuit Validation	2.58°	0.57°
Fixation Validation (±15°)	2.68°	0.45°

Table 4.2: Performance of the gaze model on different test sets for the HMD prototype.

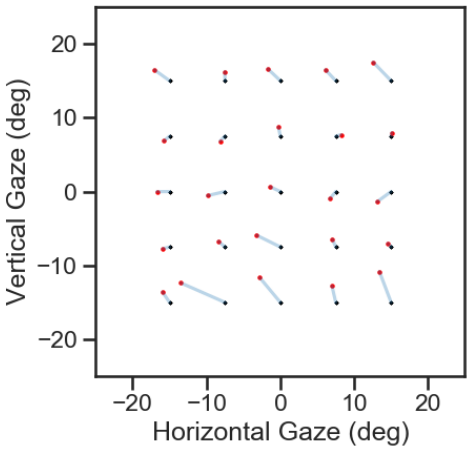

Figure 4.15: Example performance on the fixation validation task after training on smooth pursuit data. This example exhibits 2.1° mean error.

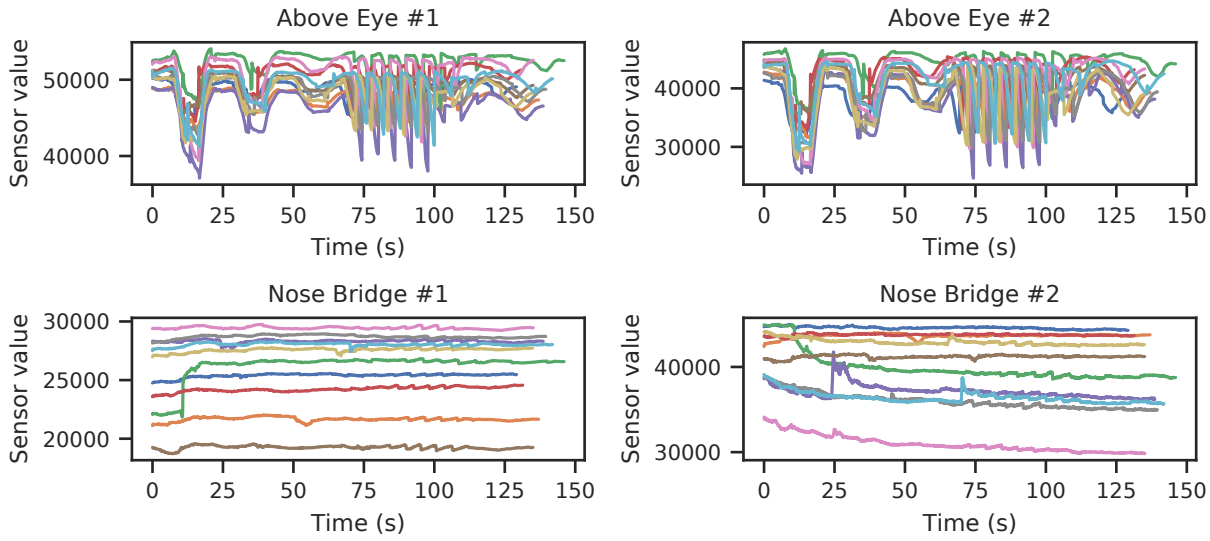

Figure 4.16: Raw output of facial-sensing photodiodes over ten smooth pursuit calibration tasks. The above-eye sensors inadvertently capture eyelid motion, but the nose bridge sensors are robust to eye movement.

4.7 Sensor shifts

4.7.1 Measuring shifts

Photodiode-based eye tracking systems are particularly susceptible to sensor shifts on the face. To quantify and account for such shifts, our HMD-device includes four photodiodes intended to measure the position of the HMD on the face. As an exploratory analysis, one user completed ten calibration trials, removing the headset after each trial to ensure the HMD is placed at a new position on the face. Figure 4.16 shows the output of the four facial-sensing diodes during each of the ten calibration runs. Ideally, these signals would be time-invariant, indicating that no slippage occurred during the calibration, and unique, indicating that each time the headset was put on, it was in a slightly different position.

The two nose bridge signals mostly fulfill these criteria, though it appears some slippage occurs early in some trials and the headset may slowly drift down on the face over time. The above-eye sensors, however, have distinct eye-movement artifacts that are correlated with vertical eye motion. It is likely that these sensors are capturing eyelid movement. This makes these sensors useful for eye tracking, but less useful for monitoring slippage, though there is a unique baseline shift for each trial. Future versions should consider alternative placements higher up on the face that are not affected by eyelid position.

4.7.2 Compensating for shifts

Despite the poor placement of two of these sensors, we conduct an exploratory analysis of this data to understand whether slippage can be compensated for. In this analysis, we compare three different approaches: 1) a global slippage model, 2) a nearest-neighbor slippage model, and 3) a baseline within-session model. The global model is a monolithic neural network trained on nine trials and evaluated on a tenth. The goal is to train a model that is simply invariant to sensor shifts. The nearest-neighbor model trains separate models for each calibration session and at runtime, chooses the model that most closely matches the observed data. The distance computation is performed using median values of the two nose bridge sensors. Ideally, this would use all four facial sensors, but the variation in the above-eye sensors make them poorly suited for this purpose. We compare 1-NN and a 9-NN model where each estimate is weighted by the distance metric. Both the global model and nearest-neighbor model would require an extensive one-time calibration, but no additional calibrations after that, even after removing the headset. Finally, the baseline within-session model simply uses a model trained on data from that session. Each method is trained using the smooth pursuit data and evaluated on the 25 point fixation task.

As a baseline, using a model trained on a random session results in 15.8° error on the fixation task. At the other end, the within-session baseline results in 2.7° error. The 1-NN model outperforms a random model, with 9.6° error and the weighted 9-NN model achieves 8.5° error. The leave-one-session-out global model performs slightly better with a 7.2° error. Interestingly, the global model error reaches the baseline of 2.7° when training data from that session is included. This suggests that a monolithic model is capable of learning sensor shifts, if it has been trained with the right data. Perhaps with more training data, the leave-one-session-out method would be improved.

Although these results show some promise in refining the results of a random calibration trial, they are not yet at a usable performance level. We believe that these results can be improved by using more appropriate facial sensors for the above-eye location and significantly increasing the amount of training data. It is possible that once a model has been trained, retraining with a new user would require just a few rounds of calibration. We also caution against generalizing too much from these results, as trials must be conducted with more users to better understand model performance.

4.8 Discussion

This chapter presented a technique for tracking the eyes using low-power photosensors. The systems presented consume tens of milliwatts of power and can be integrated into mixed reality head-mounted displays. We go beyond prior work by implementing two devices and demonstrating their performance in two small-scale studies. We also contribute new insight into the performance of photosensor-based tracking in the presence of corneal reflections that can inform future simulation techniques.

Field-of-view and eye relief present practical challenges for photosensor-based eye tracking. In our systems, we designed for $15\,\mathrm{mm}$ to $20\,\mathrm{mm}$ eye relief and at least $\pm 30^\circ$ field of view. As HMDs improve, we expect the eye relief to shrink and the field-of-view to increase. Consequently, as currently designed, the photosensors would need to move further off-axis and view the eye from the side. This is likely a difficult scenario and is additionally complicated by different head and eye geometries of users. Future designs

of photosensor-based approaches should investigate methods of combining eye tracking with the display. Efforts like bidirectional screens [56] are promising avenues of exploration.

Although the presence of strong corneal reflections complicates the calibration by producing a strong nonlinear response in a localized gaze region, it is possible that this could be a promising path toward a model-based tracking approach. Data-driven approaches, like the one presented here, are fundamentally limited by the calibration data available. If the circular corneal reflection pattern demonstrated in simulation (Figure 4.5) can be shown to generalize across a larger population, and initial results suggest it does, then it may be a fruitful signal that can be used to fit a model. For example, if a user simply performs a 1D scan, like the one shown in Figure 4.6, the location of the peak could be used to infer the geometry of the eye, using a parametric eye model [13], and the coupling between the sensor and the face. This is a problem that can be easily simulated using the right eye model.

Finally, while photosensor-based approaches are promising in terms of power and computation, there may be some precision applications for which they are simply insufficient. In these scenarios, it may be productive to combine this discrete sensor approach with traditional camera-based tracking. For example, a camera might capture the eye at $1\,\mathrm{Hz}$ to $5\,\mathrm{Hz}$ and provide a robust absolute gaze estimate while the photosensors capture the higher frequency dynamics of eye motion. In such a system, both the camera and facial-sensing photodiodes would provide useful signals for measuring sensor shifts.

5 TRACKING CONTROLLERS

5.1 Introduction

The handheld controller is a useful input device and offers advantages such as physical buttons with tactile feedback, a platform for rich haptic feedback [178, 155, 25, 194], and, most importantly, a mechanism for precise tracking. Controller tracking is classified as either 3-DoF rotational tracking (i.e. roll, pitch, yaw) or 6-DoF rotational and positional tracking (i.e. roll, pitch, yaw, x, y, z). The use of 6-DoF positional tracking enables an additional class of spatial computing applications by allowing the controller to serve as a virtual tool or a proxy for the user's hand.

Today, there is a significant divide between handheld controllers for mobile VR/AR platforms and those for high-end desktop VR systems. Mobile systems, such as the Samsung Gear VR or Google Daydream, use a small handheld controller that relies on 3-DoF inertial orientation tracking, with limited positional tracking support. On the other hand, desktop VR systems like the Oculus Rift, use a larger controller with external light emitting diodes (LEDs) for 6-DoF optical tracking. These external elements are tracked using cameras placed in the environment. Though these controllers are much larger, the positional tracking they offer results in a substantially more immersive experience compared to the simple orientation tracking on mobile VR. Efforts to remove the need for environmental infrastructure often move the tracking cameras to the head. This approach leads to additional HMD power consumption, limits cameras' line of sight, and still requires bulky tracking elements on the controller.

If head-mounted displays are to become a compelling personal computing platform with applications beyond gaming, they must support mobility and robust usage. Handheld controllers must rely on inside-out tracking to enable mobility while maintaining low power consumption for extended use on battery power. The form factor should be small enough to fit within the hand during use and in a pocket or bag when not in use. They should also support a robust set of interaction scenarios and maintain usability outside of the view of a head-mounted camera—for example, with the hands at the side on a crowded bus or under a table during a meeting. In this work, we seek to close the gap between the attractive form-factor and mobility of 3-DoF inertial controllers and the performance and usability of high-end 6-DoF controllers.

This chapter describes Aura, a novel low-power electromagnetic tracking technique to bring high-precision 6-DoF controllers to any head-mounted mixed-reality system without the need for line-of-sight, bulky tracking rings, or environmental sensors. Our proposed tracking system uses three coils embedded in a head-mounted display that each generate a unique magnetic field oscillating at 100 kHz. The generated fields induce a sinusoidal voltage in orthogonal receiver coils embedded in a handheld controller. As the user moves the controller, the signal in each receiver coil varies depending on its position and orientation within the field.

In a traditional electromagnetic tracking system, one would carefully construct a 3-axis dipole transmitter and place it far away from any other metallic objects. This significantly limits the design space of HMDs,

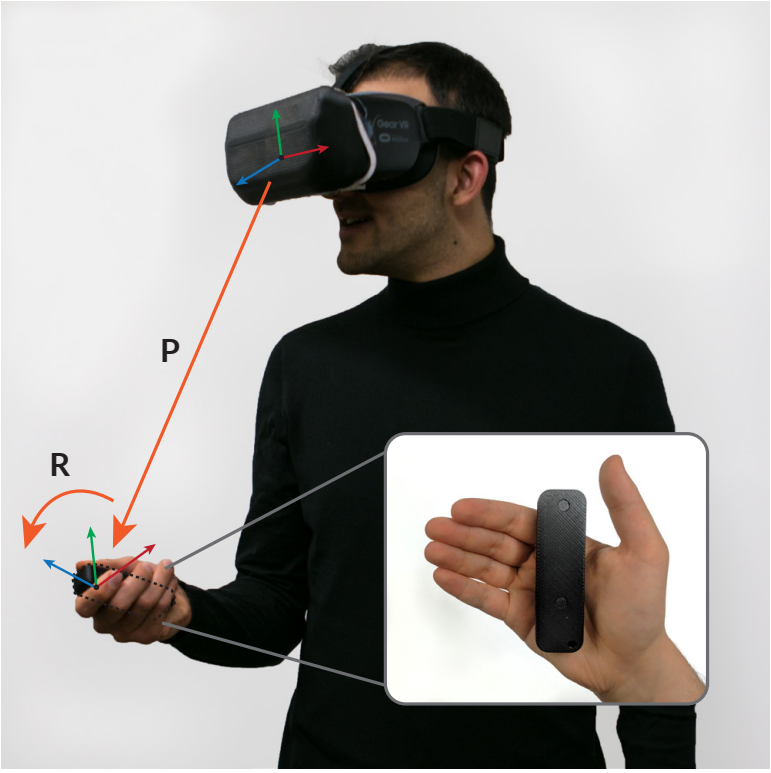

Figure 5.1: Aura is a 6-DoF electromagnetic tracking system for small, handheld controllers. It estimates the pose of the controller with respect to the head-mounted display.

which strive to be small and lightweight. Unlike traditional electromagnetic tracking systems, Aura makes no assumptions about the size, shape, or position of the transmit coils and no assumptions about nearby ferromagnetic material on the headset. This freedom enables custom coil shapes and configurations that open the design space for small and lightweight form factors. To demonstrate this capability, Aura foregoes the use of precisely manufactured orthogonal transmit coils and explicitly uses irregularly shaped coils designed to fit the contours of a Samsung Gear VR headset. The use of irregularly-shaped coils significantly complicates the pose estimation task. To solve this challenge, Aura uses a hybrid tracking approach that leverages a physics model to calibrate the sensor coils and a closed-form data-driven model using neural networks to directly estimate pose from the calibrated sensor data. While electromagnetic tracking has a rich history [131, 80, 42, 82, 102, 118, 22], to our knowledge, Aura is the first system to demonstrate 6-DoF pose estimation using irregularly-shaped, non-orthogonal coils.

Aura is inside-out; that is, it tracks the position of the controller with respect to the head. For high-end AR or VR systems with positional head-tracking, e.g. using inside-out SLAM-based tracking [55], Aura provides an upgraded controller tracking experience by enabling more robust and subtle usage without the need for line-of-sight or bulky tracking rings. On low-end VR systems that rely on inertial head-tracking, the ability to locate the hand with respect to the eyes still enables many new interactive experiences. On such devices, Aura serves as a snap-on upgrade that provides positional tracking of the controller.

Positional head-to-controller tracking allows a VR system to render the hand or virtual objects within the hand at the correct visual position and orientation no matter how the user moves their head. This capability would allow the user to perform any of the pointing-based interactions common in today's VR applications, without having to frequently recalibrate to compensate for drift. Moreover, it would allow the user to directly manipulate objects locked or loosely locked to the body. For example, a user might reach out and type on a virtual keyboard placed directly in front of them, a task that is nearly impossible with inertial tracking. We note that on low-end systems with only inertial head-tracking, the ability to directly grasp other objects located in the world is still limited by the system's head-tracking capabilities.

The following sections provide a brief overview of electromagnetic tracking techniques and design decisions we made in Aura, implementation and calibration details for the Aura system, and results from a system evaluation and characterization. Our results demonstrate that Aura can track a handheld controller with millimeter accuracy.

Specifically, the contributions of Aura include:

1. An efficient, novel, closed-form tracking algorithm that works with arbitrary transmitter coil shapes and configurations and accounts for static magnetic field distortions
2. A low-power hardware architecture for a 6-DoF handheld controller
3. A prototype implementation of the Aura system and evaluation of tracking accuracy that demonstrates a median 3D error of $5.5\,\mathrm{mm}$ and $0.8°$.

5.2 Magnetic Tracking Background

In general, magnetic tracking systems rely on two types of sources: the permanent magnet [101], which generates a static magnetic field, and alternating current (AC) electromagnetic coils [150, 149]. While permanent magnet systems are affected by the Earth's geomagnetic field, AC electromagnetic coils generate a magnetic field at a particular frequency that can be more easily measured by filtering out all other frequencies.

A typical AC tracking system consists of a three-axis magnetic field generator that produces an oscillating magnetic field and a sensor that measures the local magnetic flux density, from which the sensorâĂŹs location can be estimated. Using an alternating current to drive an electromagnet formed by a wire coil is an effective method of generating a magnetic field that oscillates at a particular frequency. According to Maxwell's equations, an electric current flowing along a wire coil will generate a magnetic field. The oscillating magnetic flux from these generator coils intersects the sensor coils, inducing a voltage of the same frequency according to Faraday's law of induction. The voltage induced in the coil is proportional to the rate of change of the magnetic flux through the surface bounded by the coil and the number of windings in the coils.

The flux through the coil depends on its orientation within the magnetic field. If the coil is aligned with the field—that is, the normal vector to the coil is aligned with the field—then the flux and the magnitude of the induced voltage will be maximal. As the coil rotates away from the field, the induced voltage decreases to zero. If the coil flips around, the voltage acquires a $180°$ phase shift, which could manifest as a negative amplitude, depending on the measurement technique.

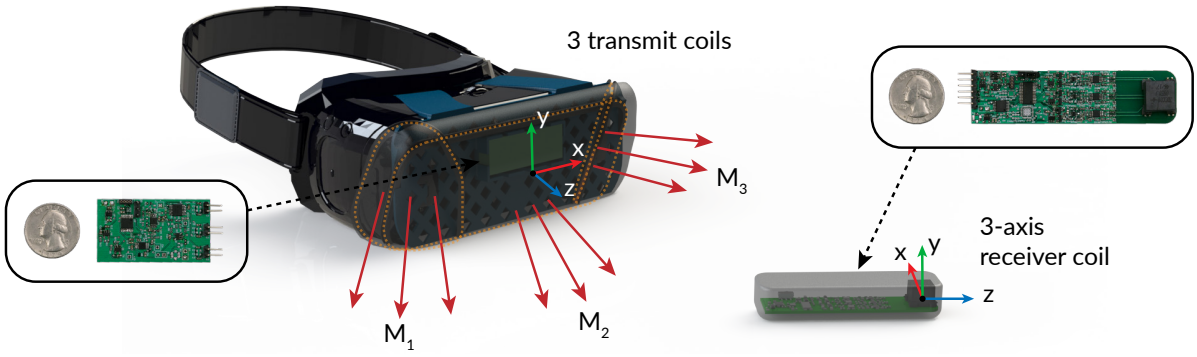

Figure 5.2: Three transmitter coils embedded in the headset produce three magnetic fields (M_1, M_2, M_3). A three-axis coil inside the handheld controller measures the fields.

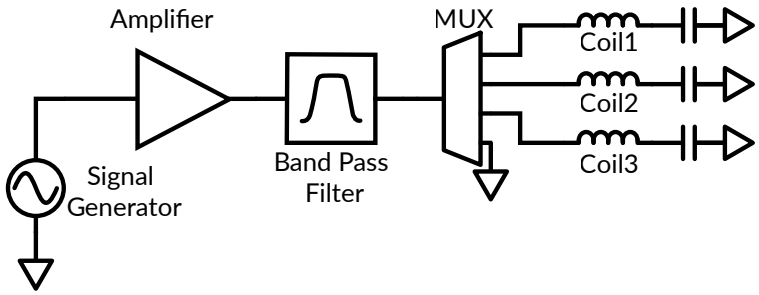

Figure 5.3: The transmitter filters the output of a waveform generator and feeds the signal through each of the transmitter coils sequentially.

To track position and orientation in real time, researchers have historically relied on tracking pose changes [131] or on iterative algorithms that find the pose which best explains the observed sensor values [123]. These systems are usually realized by approximating the magnetic sources as dipoles and, in multi-axis systems, as orthogonal dipoles. These pose estimation approaches rely on analytical or numerical analysis of forward models, which describe the magnetic field at the sensor as a function of pose. More recently, closed-form solutions [73] have been developed that analytically invert the forward models. In all of these approaches, deviations from these ideal models cause inaccuracies in the estimated pose.

These assumptions lead to traditional electromagnetic tracking systems that use large high-power transmitter coils that can be approximated as dipoles and are intended to be placed in the environment away from any metallic elements. In contrast, Aura is designed for use with arbitrary coil shapes embedded into the frame of a head-mounted display. In the Aura system, not only can the coils not be modeled as dipoles due to their shape, the presence of nearby electronics in the HMD causes static distortions to the magnetic field. Because expressing an analytic form for either the forward or reverse model is intractable,

we develop closed-form data-driven models that directly approximate the reverse model.

5.3 System Implementation

The Aura system consists of an HMD attachment that generates magnetic fields and a controller that measures the resulting fields. Figure 5.2 depicts the Aura system components. The following sections provide details of the Aura hardware and explore the capabilities and design challenges of the devices.

5.3.1 Transmitter

Aura's transmitter consists of three low-profile generator coils embedded in a head-mounted display that sequentially emit a magnetic field oscillating at $100\,\mathrm{kHz}$. Each of the side generator coils consists of 30-40 turns of 22 AWG magnet wire wound around a 3D printed ABS frame. The central coil is wrapped in an oval shape of size $20\,\mathrm{cm} \times 7.5\,\mathrm{cm}$. The inductances for the three coils are measured as $358\,\mu\mathrm{H}$, $476\,\mu\mathrm{H}$ and $279\,\mu\mathrm{H}$.

One of the contributions of our work is the use of non-orthogonal coils for the transmitter. This allows Aura to have a configuration of coils that can be fit into any HMD. The coils are rigidly mounted on a 3D-printed support structure such that the magnetic flux is directed toward the user's hand. Our implementation is designed to fit on a Samsung Gear VR (Figure 5.2) but could easily be modified to fit other HMD designs.

Figure 5.3 shows the block diagram of Aura's transmitter. The transmitter uses a programmable waveform generator (AD9833) to generate a $100\,\mathrm{kHz}$ square wave. This frequency was chosen to enhance sensitivity without approaching any of the coils' self-resonant frequencies or inducing troublesome eddy currents in nearby metallic objects. This signal is passed through a two-stage active bandpass filter (AD8616) with a Q-factor of 15.9 and gain of $0.5\,\mathrm{dB}$. The resulting sinusoid is time-multiplexed (ADG1604) and fed through each of the generator coils.

Each coil generates a magnetic field for $3\,\mathrm{ms}$ in sequence; once the cycle is completed, all of the channels are turned off for $2\,\mathrm{ms}$ to synchronize the transmitter and receiver (Figure 5.4). Because Aura cycles through each coil every $11\,\mathrm{ms}$, the resulting frame rate is $91\,\mathrm{Hz}$. An ultra-low power microprocessor (MSP430FR2100) controls the multiplexing and interfaces with the components on the transmitter board. We use passive matching networks to tune the coils impedance, resulting in improved power transmission efficiency at $100\,\mathrm{kHz}$.

5.3.2 Receiver

While the HMD-based transmitter uses hand-wound coils to generate three unique AC magnetic fields, the controller uses an off-the-shelf three-axis orthogonal receiver coil (Grupo Premo 3DCC08) to reconstruct the 3D magnetic field vector. Because the tracking system relies on interpreting the demodulated sensor measurements as field vectors, it is important to maximize precision and orthogonality in this procedure. The signal from each of the coils is fed to an amplifier (INA826) with a gain of $33.9\,\mathrm{dB}$. The resulting amplified signal is fed to a two-stage active bandpass filter (AD8616) with a Q-factor of 10.2 and a gain

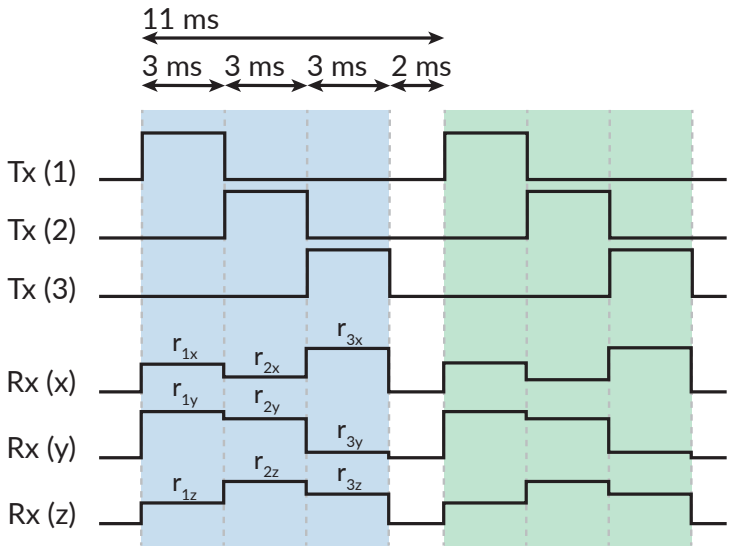

Figure 5.4: Each transmitter coil is activated for 3 ms, during which each of the three orthogonal receiver coils measures the magnetic flux.

of $33.5\,\mathrm{dB}$. Then, we use a low-noise and low-voltage drop Schottky diode network (SMS7630) in a full-wave bridge rectifier configuration to demodulate each of the channels. Finally, each channel is sampled at $4\,\mathrm{kHz}$ using the 24-bit sigma-delta ADCs of an MSP430i2031. Figure 5.5 (Top) summarizes the analog signal processing pipeline in Aura's receiver.

This method effectively captures the magnitude of the magnetic fields but does not resolve the phase of the oscillating fields. Reconstructing position or rotation would be very difficult from this approach since rotating the controller 180° along one of its axes would result in the same overall magnitude for each of the coils but with a 180° phase shift on two of the axes. Since Aura uses a three-axis receiver and each axis could be in- or out- of phase with the transmitter, there are $2^3 = 8$ possible vectors that would deliver the same sensor values. For each frame of three vector fields (one from each transmit coil), there are a total of $8^3 = 512$ possible rotation states given the same channel magnitudes.

Aura uses a low-power solution to reduce this sign ambiguity. First, comparators binarize each of the amplified signals from the receiver coils prior to rectification. The comparator outputs logic low when the amplified signal of the channel is less than its common-mode voltage ($V_{CM} = 1.2\,\mathrm{V}$) and logic high otherwise. An XOR gate estimates the relative phase between a receiver channel and a reference signal. The output of the XOR gate is low when the two signals are in-phase and high when the two are out-of-phase. One can produce a referenced signal locked to the transmitter using a phase locked loop, but to save power and simplify the design, we have chosen channel 1 of the receiver as the reference. The resulting logic signal is low-pass filtered by an RC network to remove any glitches due to phase mismatch. These digital signals are then sampled by GPIO pins of the microcontroller unit (MCU). Using this low-power approach, we reduce the ambiguity of the signs to $2^3 = 8$ possible solutions since the phase of the reference channel with respect to the transmitter is still unknown. The exact signs can be determined

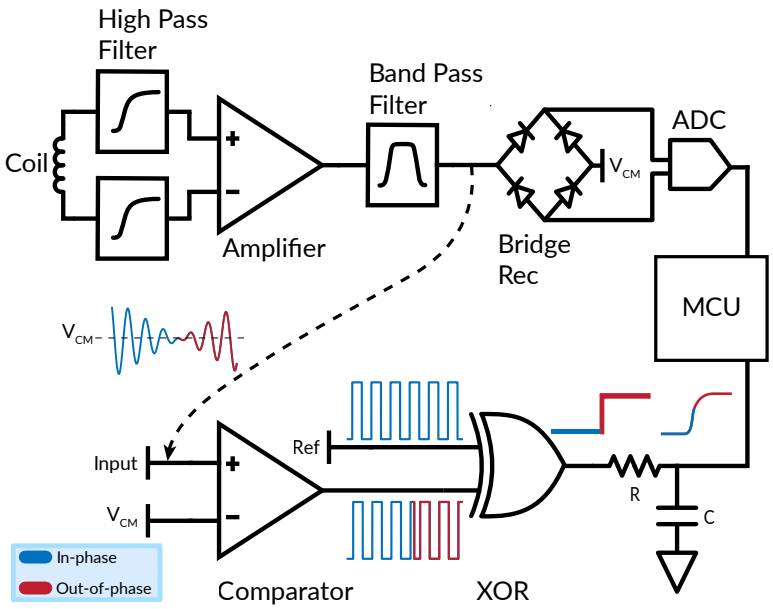

Figure 5.5: On the controller (single-channel depicted), the MCU measures the magnitude (top) and phase (bottom) of the received signal.

by placing the controller in a known start state and temporally filtering to ensure consistency over time. Figure 5.5 (bottom) summarizes Aura's sign detection capabilities.

With the amplitude demodulation and phase estimation components, the MCU now has access to a signed magnitude for each channel every 250 µs (4 kHz). This data stream contains the time multiplexed transmit signals and resembles the diagram in Figure 5.4. A segmentation step recovers each of the 9 measurements from this data stream. The segmentation algorithm will search for a local minimum to find the "off state" and synchronize itself with the transmitter. It then uses the known timing of the signal to extract the 9 mean values of each coil measurement ($r_{1\{xyz\}}$, $r_{2\{xyz\}}$, and $r_{3\{xyz\}}$). The MCU then sends the nine reconstructed signals to a PC over USB. In software, a digital second-order Butterworth filter with a 10 Hz cutoff frequency is applied to the raw signals for further noise reduction.

5.4 Calibration

In order to treat the measurements from the ADC as magnetic field strengths, a calibration procedure must be performed to account for imperfections in the signal processing chain and channel gains. This calibration is intended to be performed only once per device, i.e. through a factory calibration step. While calibration in magnetic tracking systems often refers to modeling magnetic field distortions, Aura inherently accounts for this in the tracking algorithm described in Section 5.5.

To assist in the calibration process, we construct a set of Helmholtz coils as shown in Figure 5.6. With this device, current through two parallel coils (I_1 and I_2) generate AC magnetic fields. Because of the spacing

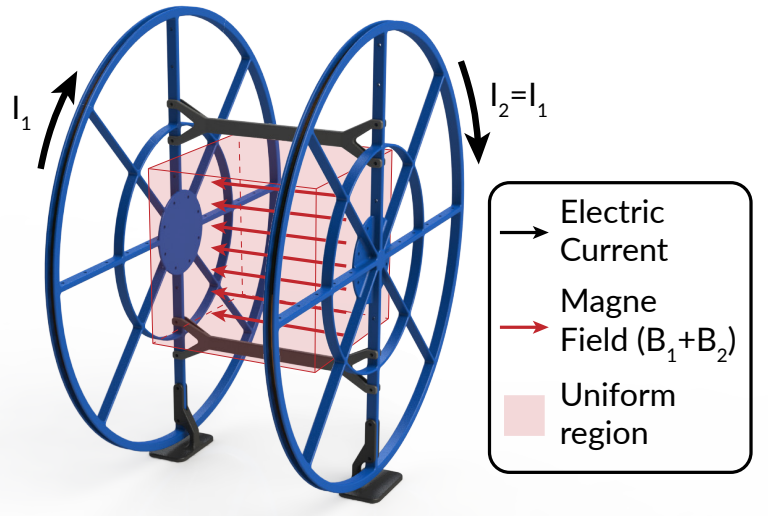

Figure 5.6: For calibration, Helmholtz coils produce a small volume where the magnetic field is insensitive to movement. The coil radius and separation distance is 15 cm.

of the coils, the tangential components of the fields cancel, and the coil pairs create a small volume with a uniform magnetic field, indicated by the red shaded region in Figure 5.6. Within that region, the magnetic field direction and magnitude are relatively insensitive to the receiver coil's position. By controlling the amplitude of the AC current through the coils with a function generator, we can precisely control the magnetic field strength within this volume.

5.4.1 Signal chain modeling

In an ideal sensor, the measurement for any axis would be linearly correlated with strength of the magnetic field along that axis. To measure this linearity, we place the Aura controller within the Helmholtz coil and use the function generator to step up the magnetic field strength linearly in small increments and record the Aura measurements. Figure 5.7 shows the observed signal and the ideal linear response. Though the signal is linear for much of the observed field strengths, significant nonlinearities were observed in the presence of weak magnetic fields.

We model these observations using three parameters: a gain term (g), Gaussian noise (n) inherent to the signal chain, and a bias term (b) due to the forward voltage drop across the diodes. Equation 5.1 summarizes how these effects influence magnetic field strength (f) to produce the Aura measurement (r).

$$r = g\sqrt{f^2 + n^2} - b \tag{5.1}$$

After collecting data from the Helmholtz coil, we use an optimization procedure (SciPy) to fit this model to the observed data. As shown in Figure 5.7, this model is a good fit for the observed data. We then invert this model to derive an expression, as shown in Equation 5.2, for the desired magnetic field strength (f)

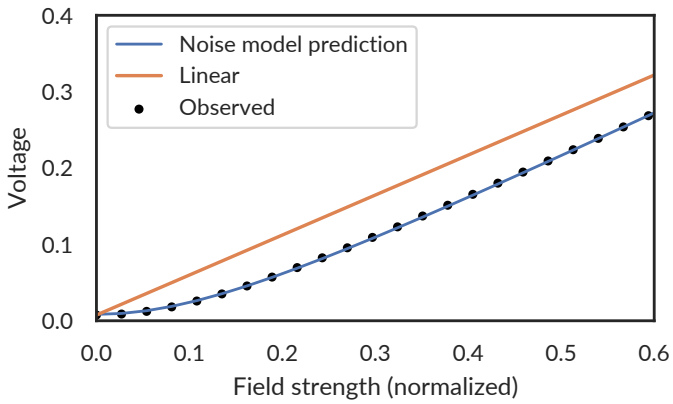

Figure 5.7: Our proposed model outperforms a linear prediction of the observed signals.

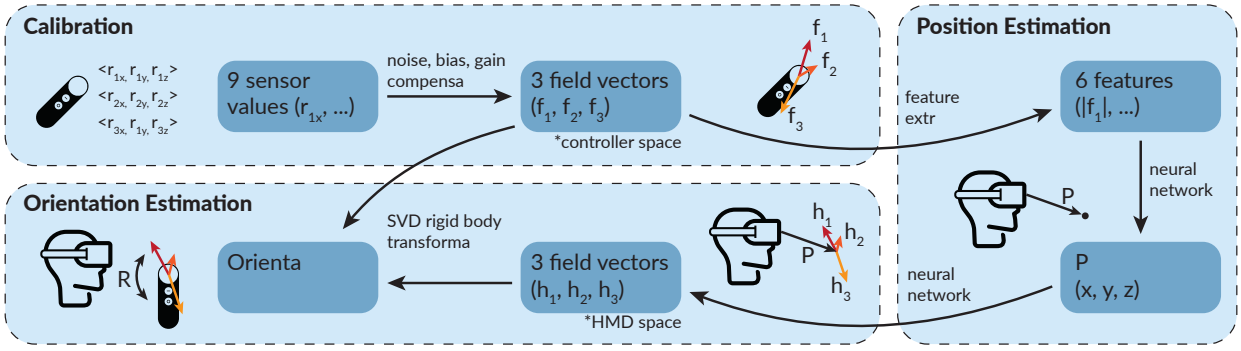

Figure 5.8: Features extracted from the field vectors are first used to estimate position. The estimated position is used to estimate the field vectors in the headset space. SVD is used to extract the orientation from these paired field vectors.

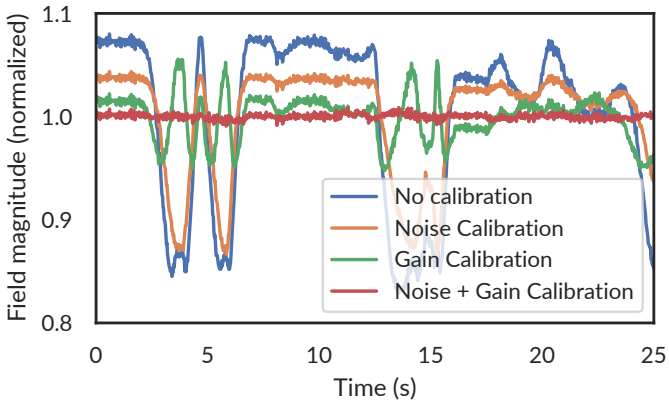

Figure 5.9: Within the Helmholtz coil, the measured field magnitudes should be constant as the controller is rotated. Without calibration (blue line), there are significant deviations in the field magnitude. As we apply different calibration steps (orange, green, red lines), the calibrated signal significantly improves.

as a function of the ADC measurements (r). We preserve the sign of the original signal as described in Section 5.3.

$$f = \sqrt{\left(\frac{r+b}{g}\right)^2 - n^2} \tag{5.2}$$

At runtime, we further improve the device performance by assuming the bias and gain terms remain constant and dynamically adjusting the noise term of this model based on the signal observed during the $2\,\mathrm{ms}$ off period of the device, when $f = 0$. The dynamic noise term is derived according to Equation 5.3.

$$n = \frac{r_0 + b}{g} \tag{5.3}$$

5.4.2 *Channel gains*

Due to component tolerances, each channel has a slightly different gain. To measure these gains, we place the controller within the Helmholtz coils and rotate it while recording data. In a properly calibrated system, the magnitude of the magnetic field measurement would remain constant as the device is rotated. We use a second optimizer to learn the optimal gains such that the magnitude of the calibrated field remains constant.

Figure 5.9 shows the observed magnitude (blue) and calibrated magnitudes after accounting for the signal chain model (orange), channel gains without the signal chain model (green), and both models (red). The constant magnitude obtained after calibration validates these models and shows that both are critical in obtaining correct magnetic field measurements.

5.5 Tracking Algorithm

Because it is infeasible to construct an analytic formulation of the magnetic fields, there is no closed-form analytic solution for the inverse problem. Instead, we rely on a closed-form data-driven solution based on machine learning techniques. Our proposed closed-form tracking approach is summarized in Figure 5.8. We treat the process of position and orientation estimation separately. At a high level, we train a neural network to regress from extracted rotation-invariant features to position. We then train a separate network to estimate the magnetic field vectors from position and use SVD to compute orientation.

Unlike other approaches, this pipeline makes no assumptions about the shape of the transmit coils or the resulting field (other than it being separable by the features we extract). This frees us from some of the restrictions of a model-based approach, which excels only when the magnetic field model accurately represents the empirical data. The training methods we describe are intended to be a one-time factory calibration.

Aura must estimate the position of the controller from the three measurements of magnetic field vectors (f_1, f_2, f_3). Because the controller can exist at the same position with any rotation, we first extract six features that are invariant to the rotation of the controller. The first three features (Equation 5.4) include the magnitude of all measured fields. As the controller rotates, the *magnitude* of each of the three fields remains constant. The second three features (Equation 5.5) relate to the angle between two fields. As the controller rotates, the direction of each measured field in controller space, f_i will change, but the angle between any two fields will remain constant. Note that in Equation 5.5, we take the absolute value of the dot product. This is because we do not have an absolute sign reference between our transmitter and receiver coils. That is, we do not know whether an ideal sensor would have measured f_i or $-f_i$. By removing the sign of the dot product, we remain invariant to this ambiguity.

$$|f_i|, \forall i \in \{1, 2, 3\} \tag{5.4}$$

$$\frac{|f_i \cdot f_j|}{|f_i| \times |f_j|}, \forall (i, j) \in \{(1, 2), (2, 3), (1, 3)\} \tag{5.5}$$

These features summarize the relative directions and strengths of the three magnetic fields. A cross-sectional slice of two of these features from the simulated dipole dataset are depicted in Figure 5.10.

5.5.1 Position

We propose the use of computationally simple models to regress to a position vector. In order to keep the models small, we split the tracking volume into four subspaces along two dimensions: 1) left ($x < 0\,\text{mm}$) / right ($x > 0\,\text{mm}$) and 2) near ($|P| < 200\,\text{mm}$) / far ($|P| > 200\,\text{mm}$). These volumes were empirically determined to balance model performance and complexity. By reducing the tracking volume for each model, we can train much smaller models than we could if the entire tracking volume were lumped together. For each of the four subspaces, we train a computationally simple neural network model with

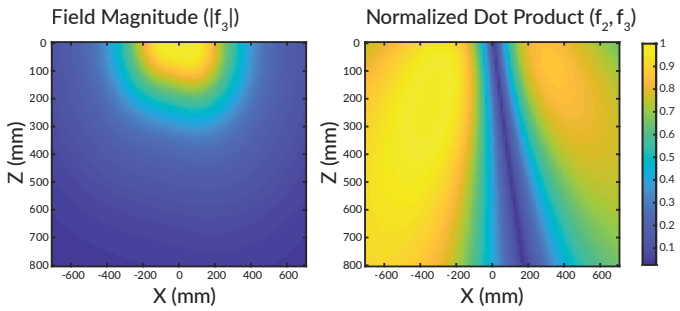

Figure 5.10: A cross-section at y = -300 mm of two of rotation-invariant features from the simulated dipole dataset. The left plot shows the magnitude of the magnetic field from the side one of the side coils, per Equation 5.4, and the right plot shows the dot product between the center and side coil, per Equation 5.5.

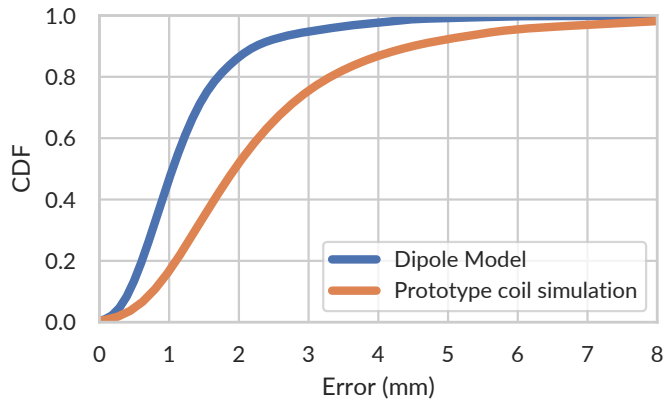

Figure 5.11: CDF of simulated tracking error for the dataset generated using dipole field models and numeric integration of the actual prototype coils.

a single hidden layer of 32 nodes to fit a function that maps the six rotationally invariant features to a 3-dimensional position vector. Training is performed using the Levenberg-Marquardt algorithm. At runtime, position estimation equates to two matrix multiplications (6×32 and 32×3), that can easily run on a mobile processor.

It must be noted that this approach adds a dependency on knowing the position before one of these four models can be chosen (to then estimate position). In this work, we use the ground truth position to pick the correct subspace model in order to validate each model separately. However, since the controller will be tracked over time, the next position can easily be estimated from temporal extrapolation or Kalman filter prediction with an IMU. This rough estimate can be used to choose the appropriate model for that frame. Additional redundancy can be added by training intermediate models that straddle two subspaces.

We simulated our approach using magnetic field simulation tools that rely on quasi-static assumptions to generate two datasets. First, we model our transmitter coils as ideal dipole models and use physics calculations to compute the magnetic field due to each of the three coils at many locations around the

head. As a comparison, we also use the BSMag toolbox [130] in MATLAB to model the specific shape of our transmitter coils and create a similar dataset of magnetic field locations.

Each dataset consists of 100,000 points generated within a tracking volume of $1.6\,\mathrm{m} \times 0.8\,\mathrm{m} \times 0.8\,\mathrm{m}$. Our model is able to estimate the position of the synthetic datasets with a median error of $1.04\,\mathrm{mm}$ for the dipole model and $1.9\,\mathrm{mm}$ for the simulation of the Aura coil design. Figure 5.11 shows the positional tracking error CDF for simulated data on the dipole model and Aura prototype model.

5.5.2 Orientation

We derive the orientation, or attitude, of the controller by comparing the measured magnetic field vectors in controller-space ($\hat{f}_1$, $\hat{f}_2$, $\hat{f}_3$), with estimates of the magnetic field vectors in head-space ($\hat{h}_1$, $\hat{h}_2$, $\hat{h}_3$), derived from a forward model. In some electromagnetic tracking systems, one can compute the magnetic fields ($\hat{h}$) as a function of position using a magnetic field model, often a dipole model [73]. In our system, we anticipate significant deviations from an ideal dipole model due to the presence of electronics around the head, the non-circular nature of our coils, and the arbitrary positioning of the coils for form-factor purposes. Instead, for a forward model, we train a separate neural network to estimate the magnetic field vector at any position, p around the headset. We again adopt a computationally-simple neural network with a single hidden layer of 32 nodes that maps the 3D position, $\hat{p}$, to three 3D vectors, $\hat{h}_1$, $\hat{h}_2$, and $\hat{h}_3$. The model is trained using a ground truth source of position and an estimate of $\hat{h}$ computed by rotating the measured $\hat{f}$ by a ground truth source of orientation. This training process is again intended to be a one-time calibration procedure.

To obtain an orientation estimate at runtime, Aura uses the magnetic field estimates and the well-known singular value decomposition (SVD) method to find the least-squared error between the two coordinate systems. For a complete review of this approach, see Sorkine-Hornung and Rabinovich [151].

5.6 Data Collection

5.6.1 Optical Motion Capture

A ground truth source for position and orientation is required to train and evaluate Aura's tracking models. To this end, we use a seven-camera Vicon motion capture system to record the real-time position and orientation of both the headset and the controller at $240\,\mathrm{Hz}$. To enable tracking, we place retroreflective spheres on both devices in known locations, as shown in Figure 5.12.

Importantly, we define the coordinate system of the controller to be the precise center of the magnetic coil so that a rotation about the origin does not change the position of the sensor. We run a one-time calibration step to find the rigid body transform between the object coordinate systems reported by Vicon and our desired coordinate system. We use the average position of the markers from Vicon and the expected marker positions from the known geometry of our system to derive this transformation, which we then apply to each frame that the Vicon system reports.

Because Aura estimates the relative pose of the controller with respect to the head, we use standard coordinate system transformations to compute this 6-DoF pose from the pose of each device in room coordinates.

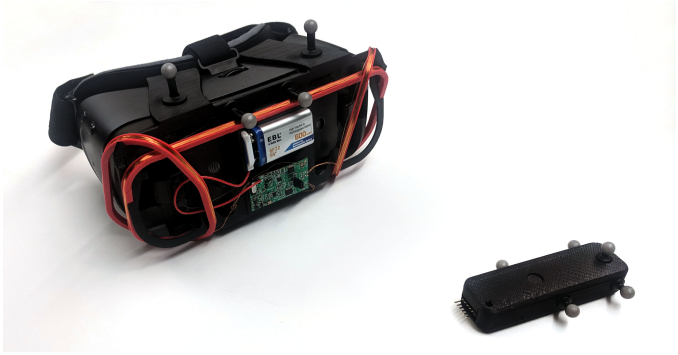

Figure 5.12: Retroreflective markers placed on the HMD and controller enable tracking with a ground truth optical motion capture system.

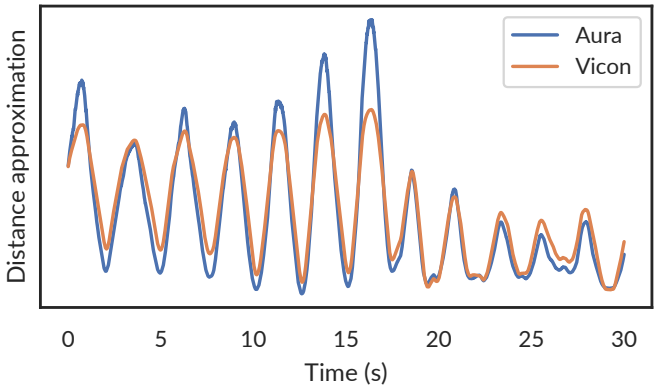

Figure 5.13: Signals used to compute alignment between Aura and the motion capture system.

5.6.2 Synchronization

Once data has been recorded from both the 91 Hz Aura system and the 240 Hz motion capture system, we must align and synchronize the two data streams. Without an electrical synchronization signal between the two, we rely on characteristics of each signal for alignment. For this, we compare the distance between the controller and headset as measured from motion capture with an approximation of the distance from Aura as specified by Equation 5.6. While this signal does not have a physically significant value, it was found to correlate strongly with the distance from the headset, as shown in Figure 5.13.

$$\frac{1}{\sqrt{\sum_{i=1}^{9} f_i^2}} \tag{5.6}$$

We first use these signals to achieve alignment at the start and end of the recorded data streams. However,

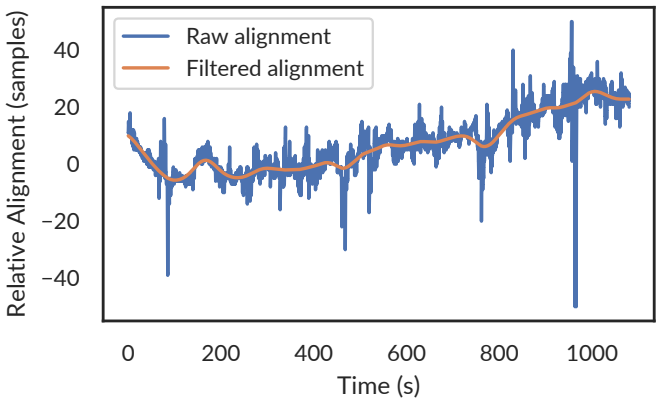

Figure 5.14: Cross-correlation between the sensor data and the distance from the headset.

due to effects like temperature changes, it was observed that the two clocks drift relative to each other over time by as much as 1 part per thousand. At a typical hand speed of $10\,\mathrm{cm/sec}$, each frame misalignment represents an additional error of over $1\,\mathrm{mm}$, so mitigating the effects of this drift is essential.

We use sliding cross-correlation between the two signals to compute a dynamic estimate of frame shift and filter this signal for smoothness. Figure 5.14 shows how the two systems drift over time. We then resample the motion capture data to align with the Aura system.

5.7 Evaluation

The main performance metrics for Aura are position and orientation accuracy, precision analysis, latency, and power consumption. We also evaluated the effects of magnetic interference on Aura's measurements.

5.7.1 Position Estimation Accuracy

2D Position Accuracy

As an initial verification of the tracking capabilities of Aura, we evaluated its positional tracking accuracy in a constrained 2D task. For this experiment, the headset was placed on a flat surface while the controller was manually moved along the surface at a mean distance of $0.5\,\mathrm{m}$ from headset. First, training data was collected by sweeping the controller across a $44\,\mathrm{cm} \times 32\,\mathrm{cm}$ area. A test set was collected by randomly moving the controller about within the same tracking area.

Figure 5.15 shows the trace from the Vicon data and the reconstructed path from Aura. The Aura system is able to track the controller with a 2D median tracking error of $1.6\,\mathrm{mm}$. A Kalman filter is used to smooth the estimate of position and reduces the median error to $1.5\,\mathrm{mm}$. Figure 5.16 shows the CDF of 2D position accuracy for the raw and filtered estimated of position.

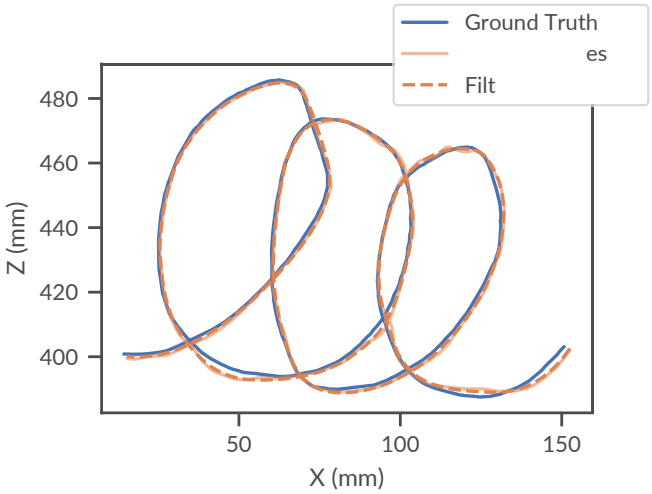

Figure 5.15: 2D positional tracking performance

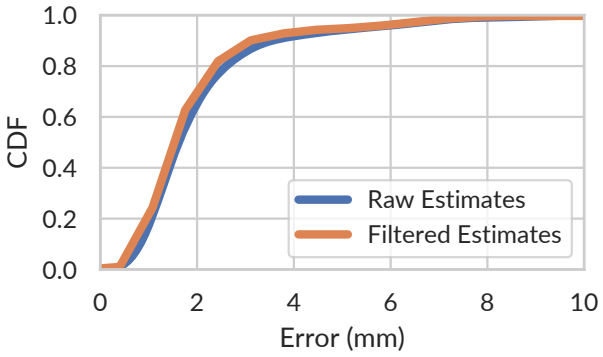

Figure 5.16: CDF of 2D positional error

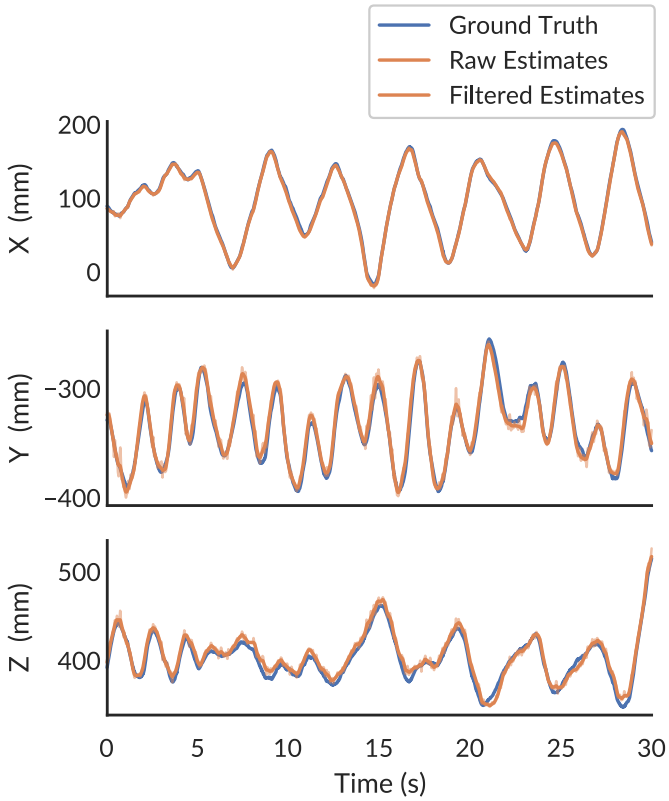

Figure 5.17: 3D positional tracking performance

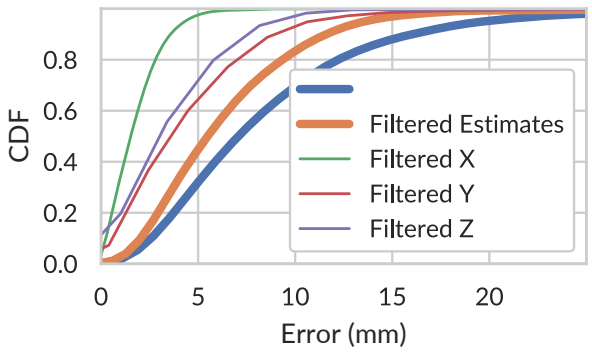

Figure 5.18: CDF of 3D positional error

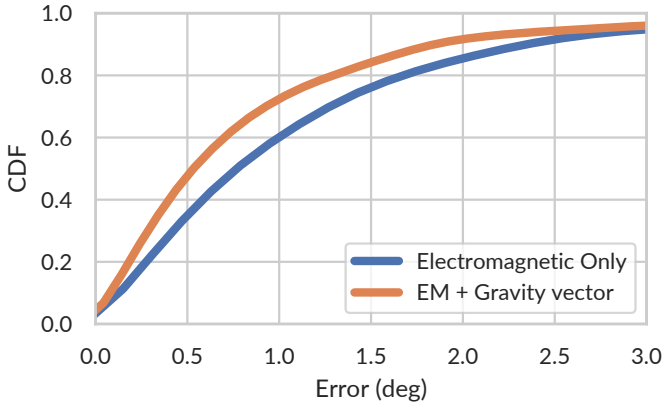

Figure 5.19: CDF of orientation tracking error

3D Position Accuracy

While the 2D test demonstrates the feasibility of this approach, Aura is intended to track handheld controllers in 3D space. For evaluating the 3D positional tracking capabilities, one of the authors wore the headset while holding the handheld controller and moving it about within arm's reach. Training data was collected for $15\,\mathrm{min}$ by systematically exploring the space in front of the user. An additional two minutes of random motion was then collected as the test set.

Figure 5.17 shows the ground truth and estimated position over a representative $30\,\mathrm{s}$ segment of the test set. Aura is able to track the controller with a 3D median error of $7.0\,\mathrm{mm}$. After applying the Kalman filter, the median error drops to $5.5\,\mathrm{mm}$. Figure 5.18 shows the CDF of 3D position accuracy for the raw and filtered estimates of position. We expect further performance gains by fusing the electromagnetic tracking with an onboard IMU.

5.7.2 *Orientation Accuracy*

We use the same dataset to train the magnetic field models for orientation estimation. Aura estimates the orientation of the handheld controller using the algorithm described in Section 5.5 and reached a median accuracy of $0.8°$. Once again, we expect significant performance gains after fusing this data source with an IMU. However, as an initial approximation, we simulate the effects of leveraging an accelerometer to recover the gravity vector. We derive this estimate of the gravity vector from the ground truth motion capture system. Adding this additional vector reference to the SVD calculation improves the median orientation accuracy to $0.5°$. Figure 5.19 shows the CDF of orientation accuracy for both Aura and the simulated approach using the added gravity vector.

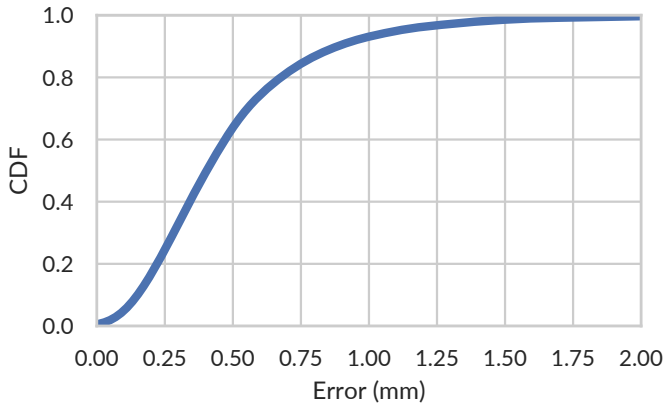

Figure 5.20: CDF of stationary measurement jitter

5.7.3 *Precision Analysis*

We also evaluated Aura's precision in its tracking estimates. To measure the precision, we used the same set up as in Section 5.7.1. However, instead of continuous motion, the device was placed in eight different locations and left motionless for a few seconds. We then compute the jitter in the estimated position while the device is motionless. The median jitter across all points is calculated to be $0.4\,\mathrm{mm}$. This calculation includes the digital low-pass filter applied to the raw magnetic signals, but no additional Kalman filtering of the position. Figure 5.20 shows the CDF of jitter.

5.7.4 *Speed and Latency*

Although the tracking analysis was performed offline, it was designed with realtime operation in mind. Like any handheld controller, we envision Aura to be used alongside an onboard IMU to capture high-speed motions. Nonetheless, we characterize the latency and speed aspects of our system.

Aura's latency is impacted by delays introduced by the analog signal chain, digital signal processing, and tracking algorithm. To quantify the impact of the analog signal chain, we place the Aura controller within the Helmholtz coil and activate the field while measuring both the current through the Helmholtz coil and the input the to Aura ADC on a oscilloscope. The time between the field turning on and the signal stabilizing is $200\,\mu\mathrm{s}$. This introduces negligible latency and validates our decision to use $3\,\mathrm{ms}$ "on" states for each coil. On the digital side, the use of interpolation to resample the magnetic signal reduces the impact of the $3\,\mathrm{ms}$ delay between channels. An additional digital filter is used to smooth data and can be set according to anticipated device usage. In our prototype, a second-order Butterworth filter with a $10\,\mathrm{Hz}$ cutoff frequency was used. Finally, we anticipate negligible latency from the tracking algorithm, which was designed with simple computation in mind. The computation consists primarily of a few small matrix multiplications, which can be performed in real time on most microprocessors.

5.7.5 Power

We measured the power consumption of both the Aura transmitter and receiver. Our measurement setup consists of a National Instruments (NI) USB 6003 data-acquisition (DAQ) unit configured for taking analog measurements in fully differential mode $-10\,\mathrm{V}$ to $10\,\mathrm{V}$ at a sampling rate of $10\,\mathrm{kHz}$. We measure the voltage drop across a shunt resistor of size $10\,\Omega$ for amount of $5\,\mathrm{s}$ and calculate the power consumed in Aura.

The handheld controller consumes an average of $13.5\,\mathrm{mA}$ ($45\,\mathrm{mW}$). Using a $700\,\mathrm{mA\,h}$ LiPo, which would comfortably fit in the controller housing, the Aura handheld controller could be tracked continuously for over two full days. Note that this does not include power consumption for a Bluetooth module or other wireless communication device.

The head-mounted transmitter consumes $29.8\,\mathrm{mA}$ average current ($224\,\mathrm{mW}$). Using a $9\,\mathrm{V}$ battery ($600\,\mathrm{mA\,h}$), the transmitter lasts for $24\,\mathrm{h}$ of continuous use. Although the current design of the transmitter uses a $9\,\mathrm{V}$ battery to operate, the Aura design is easily modified to operate at $3.3\,\mathrm{V}$. Adding more windings to the transmitter coils and reducing the operating voltage has led to prototype designs that consume only $30\,\mathrm{mW}$ on the head-mounted transmitter with similar field strengths. With this design, the Aura transmitter would last nearly four days on a $700\,\mathrm{mA\,h}$ LiPo battery.

For reference, the electromagnetic Polhemus G4 tracking system uses $5\,\mathrm{W}$ for the transmitter and $2.5\,\mathrm{W}$ for the sensor hub [125]. The HTC Vive Lighthouse base stations, an optical head and controller tracking solution, use approximately $5\,\mathrm{W}$ each, as measured by an inline power monitor. The Magic Leap One controller contains an $8.4\,\mathrm{W\,h}$ battery and is rated for $7.5\,\mathrm{h}$ of continuous use [83], suggesting a power consumption on the order of $1\,\mathrm{W}$. Aura uses at least an order of magnitude less power than these alternatives.

5.7.6 Interference

Fundamentally, electromagnetic tracking is prone to interference from nearby metallic objects. While this work does not claim any specific algorithmic contributions to account for interference, we note that the use of inside-out tracking, as opposed to outside-in tracking, significantly reduces the scope of possible interference sources. Based on our observations, significant distortion was only observed when metallic objects were placed close to either the transmitter or receiver. To quantify these effects on the Aura system, we investigated the interference from three common electronic devices that are likely to be in close proximity to the system: a smartwatch, a smartphone, and a laptop. The Aura controller and headset are placed on a flat surface as in 5.7.1. The authors then bring each interfering device from a far distance to close proximity to either the handheld controller or the HMD while recording the change on the received signals. The distance at which the signals changed by 1% was recorded. The results from this experiment are summarized in Table 5.1.

In summary, a small electronic device such as a smartwatch, has no effect as long as it is at least a few centimeters from the Aura system. Larger electronic devices must be kept further away before significant distortion is observed. As with any other electromagnetic tracking system, larger ferromagnetic materials, such as iron beams or vehicles, would have a much larger impact on the signal. While accounting for such

Table 5.1: Effects of various devices on the Aura sensor readings. Each distance represents the closest distance at which the device changes the measured signal by less than 1%.

Device	Distance to transmitter	Distance to receiver
Smartwatch (Apple Watch Series 2)	2 cm	2 cm
Smartphone (iPhone 7)	10 cm	5 cm
Laptop (2018 MacBook Pro)	20 cm	20 cm

distortions is out of scope for this work, we note that there is existing research on accounting for sources of interference [75, 74]. We also note the possibility of fusing electromagnetic tracking with inertial or optical tracking to dynamically calibrate in the presence of dynamic sources of interference.

Although not common today, it is important to consider the implications of using multiple devices within the same room. In theory, the signal from one Aura device could interfere with another if they are tuned to the same frequency. Fortunately, the strength of the generated magnetic field falls off with the cube of the distance to the transmitter. We measured the distance from the headset at which the signals fall below the noise floor of the sensor to be $1.5\,\mathrm{m}$. This indicates that there will be negligible interference between separate systems as long as they remain more than $1.5\,\mathrm{m}$ apart. For optimal operation at closer distances, the two devices should be set to different frequencies to avoid interfering.

5.8 Discussion

We demonstrate a 6-DoF tracking system capable of tracking position with a median error of $5.5\,\mathrm{mm}$ and a median orientation error of 0.8° within arm's reach around the head while using less than $50\,\mathrm{mW}$ on the controller. This approaches the performance of commercial electromagnetic tracking systems, such as the Polhemus G4, while using an order of magnitude less power and allowing optimization for form-factor. This performance does not include integration with an onboard IMU. In a production-grade system, one would use Kalman filtering techniques to fuse the electromagnetic pose estimate with inertial measurements to improve speed and precision. In our simulated result, we demonstrate that with the inclusion of a gravity vector estimate, the rotational error was reduced to 0.5°. We leave the IMU integration to future work.

While our approach demonstrates the feasibility of precise tracking, it does not fully account for all possible sensor positions and orientations. Additional robustness can be achieved by collecting data in all possible configurations, perhaps with the use of a robotic arm. For manufacturing purposes, a separate, externally calibrated sensor can be used to train the magnetic field models, as these depend only on the magnetic field, not on any specific measurement of the Aura device.

By modeling the magnetic fields empirically, Aura can operate near ferromagnetic materials, such as those found within a head-mounted display. However, this approach accounts only for static distortions to the field. Large metallic objects brought nearby the device will degrade tracking performance. Still, because Aura is an inside-out tracking system, metallic objects must be near the head or hand for distortions to

occur. Future work can explore techniques to fuse electromagnetic tracking with optical or inertial tracking to maintain accuracy in the presence of nearby distortions. Dynamic distortions caused by particular electronics within the display will likely be localized to a particular frequency and can be eliminated by carefully choosing the frequency of operation.

While Aura was designed with low-power operation in mind, we expect that additional engineering improvements can further reduce power consumption. For example, additional coil windings on the transmitter and the use of a fixed oscillator instead of a programmable function generator can save significant power on the head-mounted transmitter.

One of the advantages of our system over other electromagnetic tracking systems is the ability to use arbitrary transmitter coil configurations. While we designed the Aura prototype as a snap-on device, the transmitter coils could also be placed directly onto a PCB behind the display of a VR system or embedded within the frames of a pair of glasses. By reducing the dependence on orthogonal dipole models, we widen the design space for head-mounted computing systems with tracked devices.

Aura is designed to track a handheld controller, but by eliminating the need for bulky tracking markers, it also opens the door to other kinds of tracked objects. With small engineering improvements, Aura could be used to track accessories to improve the virtual experience, limbs for precise motion capture and avatar reconstruction, or other handheld tools, such as a pen.

Additional performance can likely be achieved by optimizing the placement and shape of the transmitter coils. The Aura coils were designed for the form-factor of a particular HMD, but our simulation results reveal a performance difference between a dipole model and a model of our coils. Leveraging simulation and optimization tools, we expect one can optimize the design for a particular use case. Despite this, the Aura prototype demonstrates reasonable tracking performance without any iteration over transmit coil design.

Aura's tracking models were trained using data collected from an optical motion capture system. However, for researchers who wish to use our system for their own projects, we anticipate that training can also be performed using commodity low-cost trackers, such as the HTC VIVE Tracker.

5.9 Conclusion

Aura is a head-mounted inside-out tracking system for handheld devices. We demonstrate a novel low-power architecture that enables precise tracking without the need for external infrastructure, line-of-sight, or bulky tracking markers. In an evaluation with an optical motion capture system, we demonstrate Aura's ability to track a controller with a median error of $5.5\,\mathrm{mm}$ and $0.8°$ within arm's reach. We hope that our system enables increased adoption of mobile spatial computing systems.

6 HAPTICS FOR HANDHELD CONTROLLERS

6.1 Introduction

In addition to precise tracking, handheld controllers offer a platform for haptic feedback. However, the ability of such devices to render the sense of touch on controllers is lacking. Haptics on handheld controllers are limited to vibrotactile stimulation, which is typically used for notification or binary touch events. This lack of cutaneous cues limits the user's ability to feel contact with a surface and to explore its texture and shape.

For more nuanced haptic rendering, researchers have developed finger-mounted haptic devices [112, 147, 128, 23, 111, 189, 47, 145], glove-based exoskeletons [36, 39, 31, 14], and robotic arm solutions [6, 109, 97, 51, 78, 168] to render various haptic sensations. These devices either require users to mount or wear additional devices or require expensive robotic arms with a limited range. Researchers have also explored the use of handheld devices for haptic rendering [12, 194, 129]. Such devices are convenient to use and they are likely more compatible with existing VR systems because they can replace the functionality of existing controllers. However, previous controller-based devices have only focused on rendering a single haptic stimulus (e.g. normal forces or weight distribution).

This chapter presents Haptic Revolver, a reconfigurable handheld haptic controller for virtual reality. The device uses an actuated wheel underneath the fingertip that moves up and down to render touch contact with a virtual surface and spins to render shear forces and motion as the user slides along a virtual surface. The device's wheel is interchangeable and it can contain a variety of physical haptic elements, such as ridges, textures, or custom shapes (Figure 6.1). These haptic features on the wheel's outer surface provide different sensations to the user as they explore the virtual environment. Because the device is spatially

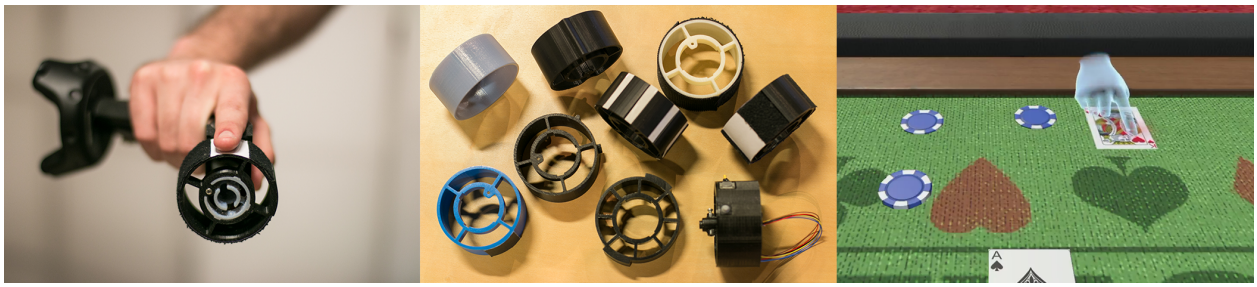

Figure 6.1: (left) Our Haptic Revolver device uses a wheel that raises and lowers and spins underneath the fingertip to render various haptic sensations. (center) The haptic wheels are interchangeable and can be customized to render arbitrary textures, shapes, or interactive elements. (right) Wheel features are spatially registered with the virtual environment, so the user can reach out and feel virtual surfaces.

tracked, these haptic elements are spatially registered with the virtual environment. As the user explores a virtual environment, our rendering engine delivers the appropriate haptic element underneath the finger. For example, in a virtual card game environment, when a user touches a card, a poker chip, and a table, the device rotates the wheel to render the appropriate texture underneath the fingertip. As the user slides along one of these surfaces, the wheel moves underneath the finger to render shear forces and motion.

Unlike other haptic devices [12, 23, 31], which always maintain contact with the finger, our Haptic Revolver device can selectively contact the finger. When a user touches a virtual surface, the wheel rises to contact the fingertip. Because the haptic wheels on our device are interchangeable, Haptic Revolver can generalize to many applications. Applications can use custom wheels with the necessary haptic features. For example, a virtual petting zoo game might use a wheel containing various textures while a virtual cockpit environment might use a wheel with input elements such as buttons and switches.

The following sections describe the design and implementation of our device, the techniques we use to render an arbitrary scene, and results from two perceptual studies that informed these decisions. Our results show that we can change the wheel speed and direction to render arbitrary scenes without compromising realism and support our technique of rendering 2D motion with a single wheel. We also show several example applications that highlight functionality of our device and qualitative feedback from users.

Specifically, the contributions of Haptic Revolver include:

1. The design of Haptic Revolver, a handheld VR controller that renders touch contact, pressure, shear forces, textures, and shapes using a rotating wheel beneath the index finger;
2. Interchangeable haptic wheels that can be used to render surface features and techniques to haptically render any scene using an arbitrary wheel;
3. The results of two perceptual user studies that inform the design of our haptic rendering strategies.

By combining the fundamentals of touch contact, pressure, and shear rendering with the flexibility of haptic wheels that support arbitrary shapes and textures, Haptic Revolver enables more accurate haptic rendering for virtual environments.

6.2 Haptic Revolver Implementation

To render contact and motion on the fingertip in a compact form factor, we chose to use a wheel that raises and lowers and rotates in response to its position in the virtual environment. In the following sections, we describe the hardware and software of our system as well as the design of interchangeable haptic wheels to deliver custom haptic sensations.

6.2.1 Mechanical Design

We arrived at the design of Haptic Revolver through an iterative process. Each design, shown in Figure 6.3, improved the functionality and ergonomics of the device. Our final design, shown in Figure 6.2 has two degrees of freedom, each of which are actuated by a motor. A servo motor (Hitec HS-5070MH) raises and lowers the wheel assembly along an axis positioned along the grip of the controller. The wheel assembly is positioned along the axis of the index finger and consists of a 12 V DC motor (Faulhaber 1524_SR) housed

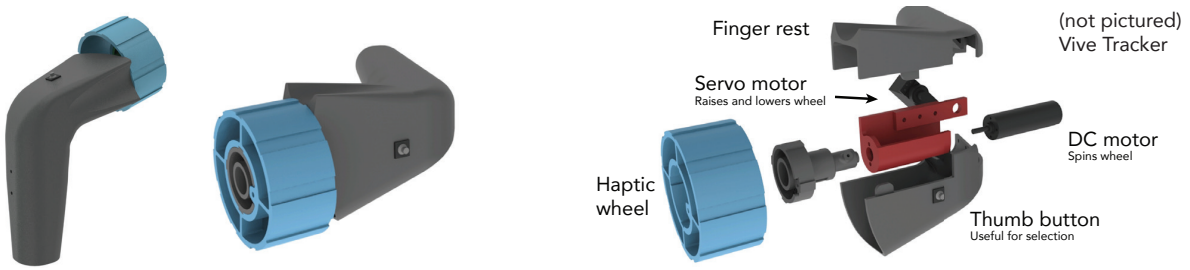

Figure 6.2: (left) 3D model of Haptic Revolver with a textured wheel attached. (right) Exploded view showing internal components. A servo motor raises and lowers the wheel, while a DC motor spins the wheel to render motion and shear forces. A button on the side allows the user to select objects and navigate. The VIVE tracker (not shown) enables 6-DOF spatial tracking of our device

Figure 6.3: Examples of previous iterations of our device. This iterative design process led to important design features of our current device.

in a 3D printed mount. The motor includes a 19:1 gearhead and a 4096 count 2-channel magnetic encoder. A wheel mount on the end of the wheel assembly allows custom wheels to be easily attached. With this gear ratio, the motor can spin at $180\,\mathrm{rpm}$, which corresponds to a linear motion underneath the finger of $565\,\mathrm{mm/s}$, assuming a $60\,\mathrm{mm}$ wheel diameter.

The controller is designed so that the index finger rests in a groove along the wheel axis. This lets the finger naturally rest on the surface of the wheel while preventing horizontal motion as the wheel spins. For improved ergonomics, the axis of the finger wheel assembly is offset from the grip handle by 110°.

A tactile button on the side of the device is used for selection and navigation and is in easy reach of the thumb during normal operation. If desired, there is room on the side of the controller for additional input elements, such as buttons, a joystick, or a touchpad. For positional tracking we attach the HTC VIVE Tracker to the end of the device, as shown in Figure 6.1 (left). This device integrates easily with the HTC VIVE head mounted display and Lighthouse tracking system and reports a 6-DOF position and orientation.

Although we envision this device eventually being wireless, for simplicity, we use the device with a power and data tether. In our current implementation, both the power supply and electronics are external to the device. We note that there is enough room within the device grip for eventual placement of electronics and

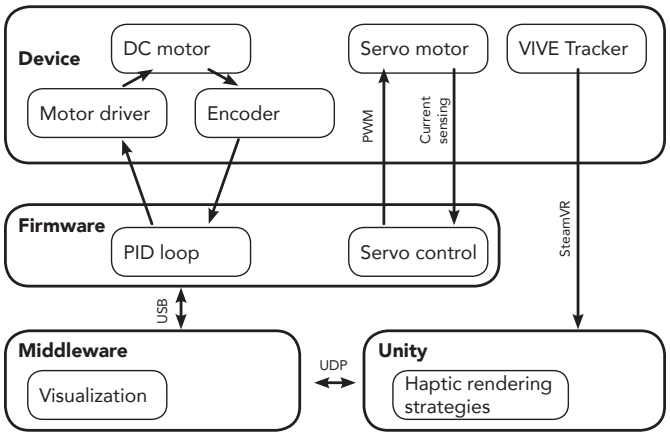

Figure 6.4: Architecture of the Haptic Revolver device and software stack. The device is powered by a PSoC, which sends commands to the two motor drivers and communicates with the PC via a USB serial connection.

a battery. The device, including the VIVE tracker, weighs 237 g, which is comparable to a VIVE Controller (205 g). Table 6.1 shows additional device specifications.

Weight	237 g
Max wheel speed	180 rpm
Wheel diameter	60 mm
Max motion under finger ($\alpha = 1$)	565 mm/s
Max force against finger	3.35 N
Typical power consumption	1.25 W
Peak power consumption	2.5 W

Table 6.1: Mechanical and electrical specifications of our device

6.2.2 Software Architecture

The software architecture is summarized in Figure 6.4. The device is controlled by firmware running on a Cypress Programmable System on Chip (PSoC) 5LP. A PID loop on the PSoC turns the wheel to a specified rotation from a known starting angle using an external motor driver (DRV8871). The PSoC also drives the servo through pulse-width modulation (PWM). The two motors are each powered by separate step-up power regulators (Pololu U3V50F*). In order to measure the force exerted by the finger against the wheel, the PSoC also monitors the voltage across a 1 Ω shunt resistor in series with the servo motor. The PSoC interfaces with a PC using a USB serial connection running at 115 200 baud.

A Python middleware layer on the PC handles communication with the device, visualization and logging, and communication with the VR application. The application layer is built with the Unity 3D game engine.

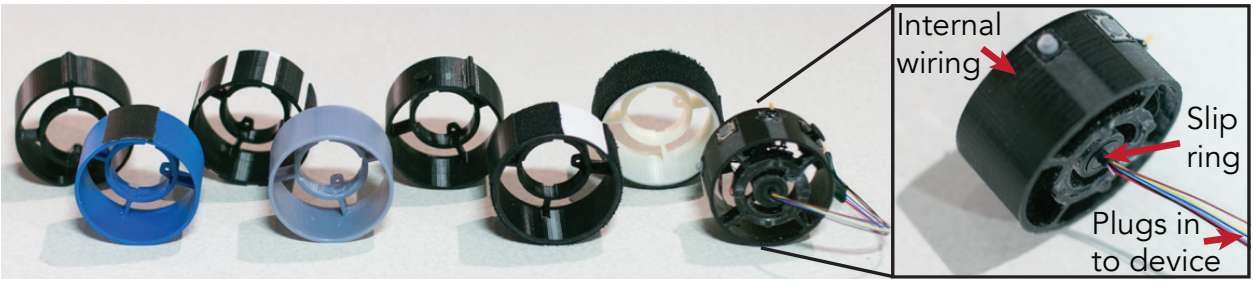

Figure 6.5: Interchangeable haptic wheels allow applications to customize the haptic experience with various shapes and textures. (right) Wheels with active components use a slip ring in the wheel mount to wire the electronics in the wheel back to the device.

Our Haptic Revolver rendering engine in Unity determines the ideal wheel configuration and streams the desired settings to the Python middleware using a socket connection.

6.2.3 Interchangeable Haptic Wheels

While a simple plastic wheel can simulate touch contact and motion of the finger, there are many applications that would benefit from custom textures or shapes placed on the wheel to match elements in the virtual environment. Haptic wheels, such as those shown in Figure 6.5, are designed to slide onto the wheel mount and can be 3D printed or manufactured from other materials. The ability to customize wheels allows us to render certain objects with much higher fidelity. For example, a simple plastic wheel can be easily augmented by affixing materials with unique textures, such as cloth, rubber, or paper, which correspond to particular objects in the virtual scene. Textures such as bumps and grooves can also be printed directly into the wheel itself to render various surface textures. Coarser shapes printed on the wheel can simulate larger features in the scene. For example, a wheel with a raised region, such as the one shown in Figure 6.6 can be used to render edges. With such a simple wheel, a user can feel the boundaries of a physical button or feel when the finger slides off the edge of a surface. Other custom shapes can be designed to match specific objects in the scene. In a sculpting application, for instance, appropriate shapes on the wheel can allow the user to feel a tool beneath their finger during use.

Although many sensations can be rendered using passive wheel elements alone, additional functionality can be achieved with active wheels containing electronic components. For example, an active wheel can include input elements, such as buttons and switches that directly map to virtual widgets and add interactivity. Components such as Peltier elements can be added to the wheel to create additional haptic sensations that can be controlled dynamically.

To deliver electrical contacts onto a spinning wheel, we designed a wheel containing a slip ring, as shown in Figure 6.5 (right). Up to 12 wires attach to elements on the wheel, pass through the slip ring and out the front of the device, and plug in to a port on the bottom of the device. We created one such wheel with input elements that include buttons, a switch, and a joystick to enable interaction with different virtual widgets. We envision future designs of our device to include a through-bore slip ring in the wheel mount itself. In this design, electrical contacts to the wheel would be made through a physical connection along

the outer face of the mount. This would move all the electrical components to the interior of the device and enable custom active wheels without the overhead of additional wiring.

6.3 Haptic Revolver Rendering Engine

Because each wheel has a limited surface area, haptic elements on the wheel will likely not precisely match the size and position of elements in the virtual environment. We developed a rendering engine to analyze the scene, hand trajectory, and wheel configuration and determine how to control the device. As there may be competing goals within the scene, the engine operates by constructing and resolving a set of constraints to take into account dragging motion and the desired orientation of the wheel. At each frame (roughly $90\,\mathrm{Hz}$), we raycast beneath the finger to determine the nearest collision with a haptic surface. To minimize jerky movements of the controller, we smoothly raise the wheel as the finger approaches a surface. We use similar penetration compensation techniques as described by Benko et al [12]. As the hand penetrates the virtual surface, we raise the wheel even further to provide pressure feedback to the user. Visually, the hand remains at the same height.

From the predicted penetration point, we scan left and right to determine which other haptic elements are nearby. If the user is making contact with a surface, we add shear constraints to move the wheel along with the user's motion. If no contact is made, we allow the wheel to spin quickly to ensure the constraints are met. If other haptic elements are nearby, we add positional constraints to ensure that the features on the wheel align with the virtual elements. In the constraint resolution step, we resolve any shear constraints with positional constraints to arrive at a desired wheel orientation.

To illustrate this process, consider the simple virtual scene shown in Figure 6.6 used with a wheel containing a small raised region, shown in black. As the user hovers over the blue surface (left), we ensure the correct texture (shown in blue on the wheel) is placed beneath the finger. We also make note of the nearby raised surface (shown in black) and add that as a constraint. As the user moves closer to the raised surface (center), the edge constraint is given higher priority and the feature on the wheel is brought close to the finger. If the user were making contact with the surface, they would feel the edge in the correct location. Finally, as the user moves onto the raised region (right), we impose two constraints, one for the edge in each direction. This effectively scales the gain of the wheel rotation so that the edges are placed in the correct spot, regardless of the size of the raised region.

When rendering shear forces during contact with a surface with no other constraints, a natural option is to spin the wheel such that the linear movement under the finger matches the linear movement in virtual space. In practice, this leads to quickly running out of room on the wheel before another feature arrives. To balance the realism of the dragging motion with practical constraints, we choose to reduce the wheel gain, α, to 0.6. For every $1\,\mathrm{cm}$ of finger motion, we spin the wheel such that it travels $0.6\,\mathrm{cm}$ beneath the finger. This value was chosen based on the results of a perceptual study described later. It represents the smallest gain before significant reductions in realism are observed.

Because our device uses a wheel, it inherently renders motion in only one dimension (horizontally). While this is appropriate in many scenarios, it would be ideal to support motion in two dimensions (horizontally and vertically). Because we noticed that users were insensitive to the direction of motion under the finger when the surface is smooth, we simulate vertical motion by simply spinning the wheel horizontally.

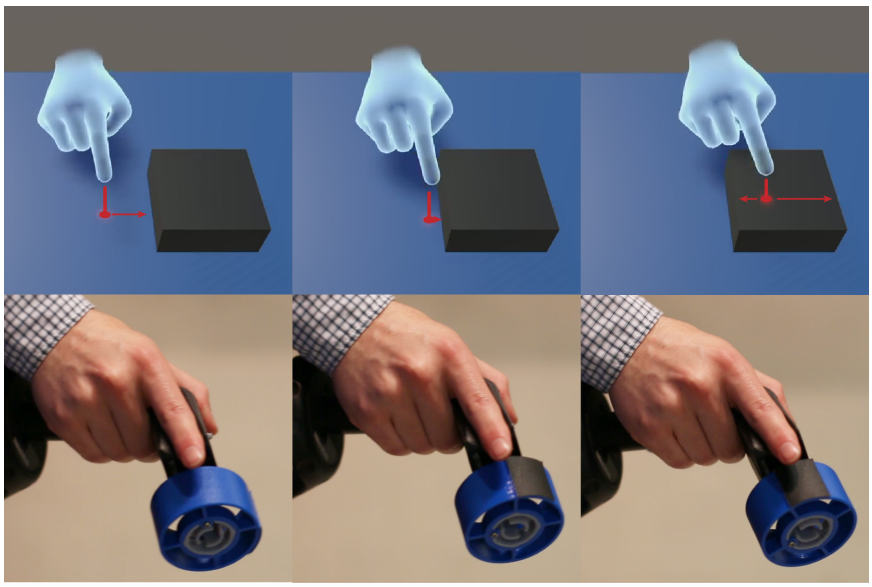

Figure 6.6: (left) When a user hovers over the blue surface, the rendering engine places the appropriate wheel surface under the finger and begins to track the nearby edge of the black surface. (center) As the user approaches the edge, the rendering engine positions the wheel so that the edge approaches the finger. (right) While hovering over the smaller black surface, the rendering engine adjusts the gain of the wheel so that the two edges are rendered correctly.

Although this is orthogonal to the actual direction of motion, prior work supports the feasibility of this illusion [190]. In this mode, we choose the direction based on the horizontal component of velocity and allow the wheel to switch directions only when there is a sudden change in hand direction or when the hand comes to a stop. We evaluate the efficacy of this rendering technique for different types of tracing behavior in Study 2.

6.3.1 *Rendering with Custom Wheels*

To allow new wheel designs to be easily added without modifying the scene, we created a simple wheel description specification, implemented as a JSON file. This file contains a list of features on the wheel and where they are located. The rendering engine uses this wheel description file to determine how to control the device. As the finger approaches a haptic element in the scene, the virtual object reports its desired haptic feature, such as a soft texture. The rendering engine finds the appropriate element on the wheel and turns it according to the constraint resolution steps. If a desired haptic element, such as a soft texture, is not present on the wheel, the engine will fall back on a suitable replacement feature, such as a smooth texture. Figure 6.7 shows an example wheel description file for the wheel shown in Figure 6.12.

As additional haptic features are added to the wheel, the amount of wheel space available for any one feature is reduced. This can make it more difficult to render the sensation of dragging along a surface, as the finger will quickly collide with an additional haptic element. To address this challenge, we developed

```
{
  "name": "CasinoWheel",
  "features":[
    { "start": 0, "stop": 47.7, "height": 1,
      "texture": "hard", "name": "poker"},
    { "start": 47.7, "stop": 90, "height": 0,
      "texture": "soft", "name": "felt_small"},
    { "start": 90, "stop": 135, "height": 0,
      "texture": "paper", "name": "card"},
    { "start": 135, "stop": 360, "height": 0,
      "texture": "soft", "name": "felt_large"}
  ]
}
```

Figure 6.7: An example wheel description file used by the rendering engine. This file describes the wheel shown in Figure 6.12.

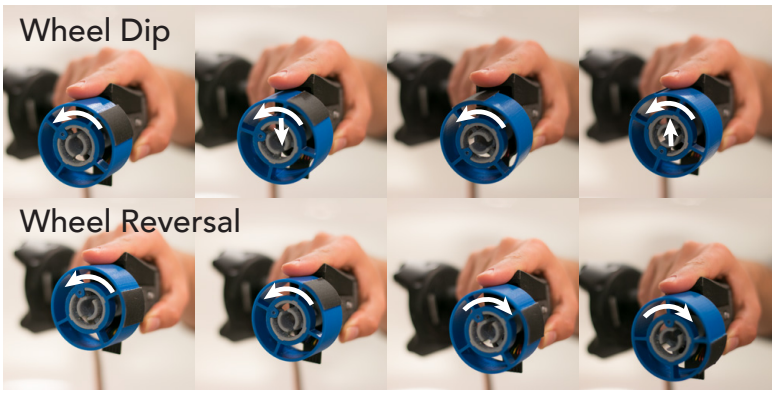

Figure 6.8: (top) As the finger approaches an obstacle (indicated here by the black region on the wheel), the dip strategy causes the wheel to lose contact with the finger while an undesired feature remains under the finger. (bottom) In the reversal strategy, the wheel begins to rotate in the opposite direction when an undesired feature is encountered.

several strategies for hiding undesired features on the wheels. These strategies are illustrated in Figure 6.8.

Wheel Dip: Our first strategy simply lowers the wheel just before a feature would approach the finger. We ease the wheel position in and out to create a smooth transition and prevent jerky behavior. This strategy has the effect of causing the finger to lose touch with the wheel for a short period of time. We can actually shorten the time without contact by accelerating the wheel over the undesired feature once it has lost contact with the finger. This can reduce the amount of disturbance caused to the user.

Wheel Reversal: As an alternative strategy, we simply reverse the wheel direction before a collision occurs. While dragging along a surface using a wheel with other elements, the wheel will rotate back and forth to render the appropriate shear motion while keeping the finger on the correct region of the wheel, effectively hiding the other haptic features. This behavior is supported by our findings from Study 1, which revealed that rendering motion in the opposite direction has little impact on perceived realism. Although the overall direction of motion may not be noticeable, the act of switching directions does cause a noticeable shear

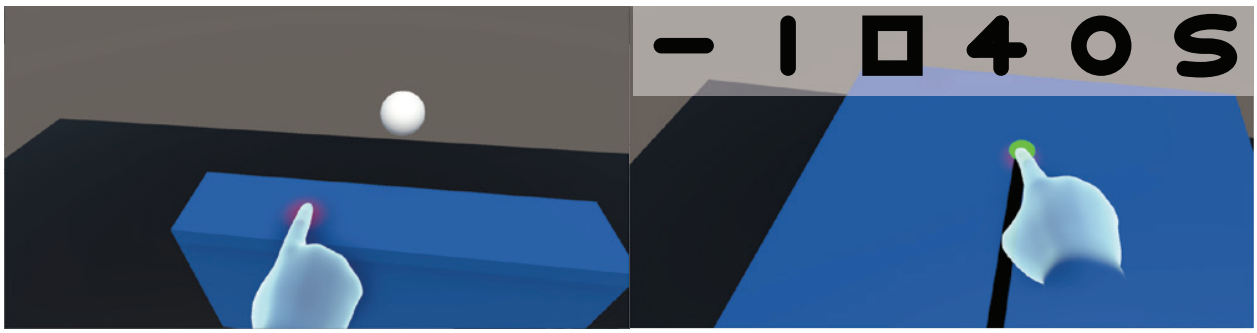

Figure 6.9: (left) In the first study, users slid their finger horizontally across a surface. (right) In the second study, users traced a path on a surface. (right, inset) The six paths used in the second study

against the fingertip. While this is not entirely unavoidable, we can reduce the frequency of such reversals by reducing the wheel gain, α. This method is also less noticeable if the user grips the controller more tightly, which reduces horizontal finger wobble during a direction change.

Because these two methods each have their advantages, we use them both in our rendering engine, depending on the circumstances. When a user is sliding over a surface that has a large physical size on the wheel, we use the reversal technique. We observe that this technique causes less disturbance overall because it never loses contact with the finger. If the physical region is small or the wheel needs to change orientations to accommodate other constraints, we use the wheel dip technique to skip ahead to the desired wheel region.

6.4 Evaluation

To evaluate the fundamental haptic capabilities of Haptic Revolver, we conducted two studies to understand how wheel parameters impact the realism of the haptic rendering. In the first study, we measure the impact of the wheel speed gain and direction. In the second study, we explore simulating motion in two dimensions using a single wheel. These studies helped inform the design of our haptic rendering techniques. We recruited 12 right-handed participants (10 male, 2 female), age 18 to 48, to participate in both studies. Participants were instructed about the nature of the study and given a short overview of the Haptic Revolver device. Participants then put on an HTC VIVE head-mounted display and held our Haptic Revolver device in their right hand and a standard HTC VIVE controller in their left hand. All studies were conducted while the participants were standing. Participants briefly explored a demonstration scene where they were able to touch and swipe on a virtual object before the studies began. Each study took approximately 20 minutes and participants were compensated with an $8 meal coupon at a nearby cafeteria for their time.

6.4.1 Study 1: Rotation gain

In this study, we sought to understand how the wheel speed gain and direction impact the perceived realism of the haptic rendering. When the user's hand moves a distance of x to the right, the wheel spins such that a distance of αx has passed underneath the finger to the left. To most closely match reality, we would set the gain, α, to 1. However, in some cases, it is useful to modify the gain to move the wheel to a desired orientation more quickly or more slowly. We also wanted to explore how important it is to spin the wheel in the correct direction. We hypothesized (**H1**) that a one-to-one mapping from virtual motion to wheel motion (a gain of 1.0) would be most realistic. We further hypothesized (**H2**) that users would prefer that the wheel spins in the natural direction ($\alpha > 0$), but that they would prefer a reverse spin ($\alpha < 0$) to no spin ($\alpha = 0$).

To test this, we asked users to swipe their finger along the length of a $50\,\mathrm{cm}$ wide virtual surface under 17 different gain settings from $\alpha = -1.6$ to 1.6 in increments of 0.2. Participants began the trial by positioning the tip of their finger within a small sphere on the surface. After exploring the surface for several seconds, participants ended the trial by moving their finger within the bounds of a second sphere, positioned just above the surface. We then asked the user to rate the haptic realism by responding to an on-screen prompt. The prompt asked users "*How closely did the haptic rendering match your visual impression of the scene?*". All responses were collected on a 5-point Likert scale (1-not at all realistic, 5-highly realistic) by pointing at the desired response on screen and clicking with the VIVE controller in the left hand. Each block consisted of a single repetition of each gain setting presented in a random order. Participants completed three blocks each.

Results

Figure 6.10 shows the average rating across participants for each condition. In this plot, we normalize responses such that the highest and lowest responses for each participant become '5' and '1', respectively. Participants reported the lowest realism score for $\alpha = 0$, or when the wheel never moved. Realism scores increased as the wheel gain increased, but leveled off around $\alpha = 0.6$. Wheel direction had little impact on realism as shown by the symmetric nature of the graph. When asked after the study, only two users even noticed that the wheel was spinning in the reverse direction some of the time. This is consistent with prior work, which found that the direction of skin deformation had little impact on realism [190]

These results support several aspects of our rendering techniques. Most importantly, it suggests that as long as the wheel speed gain is at least 0.6, the gain does not matter much. This allows for some flexibility to spin the wheel faster or slower and accommodate other constraints. Second, these results validate our approach to avoiding features by reversing the wheel direction. Finally, these results do not conflict with our decision to render vertical motion with horizontal motion under the finger.

6.4.2 Study 2: Vertical movement

Though Haptic Revolver only spins in one dimension, it is important to explore whether we could effectively render motion in two dimensions. Since the results of Study 1 suggest that spinning in the opposite

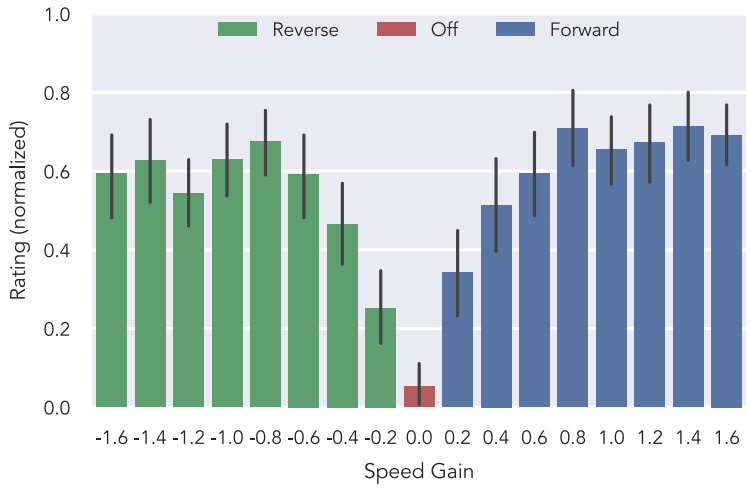

Figure 6.10: Results of the first user study showing mean realism ratings across participants as a function of the wheel speed gain. The error bars show a 95% confidence interval. A negative gain indicates the wheel was spun in the opposite direction.

direction had little impact on perceived realism, we hypothesized that spinning in an orthogonal direction would also have little impact.

To explicitly test this, we displayed a path on a flat surface and asked users to trace the path in the forward and reverse directions. Each experimental block consisted of six paths and five wheel behaviors for a total of 30 trials, which were presented in a random order. Participants completed each block twice. To explore the effect of path shape, we chose paths (Figure 6.9, right) that include a combination of horizontal and vertical motion as well as a mixture of sharp edges and curves. Paths were scaled to fit within a $25\,\mathrm{cm}$ by $25\,\mathrm{cm}$ square.

In addition to modes that render wheel motion in the horizontal direction (Motion 1D) and in both directions (Motion 2D), we introduce three other baseline conditions. Our five wheel conditions are:

- **Motion 1D**: As the finger moves horizontally, the wheel spins with a gain of $\alpha = 1$. As in the previous study, moving vertically causes no change to the wheel.
- **Motion 2D**: Similar to Motion 1D, except the wheel also spins when the finger moves vertically. After a sudden change in direction or when the finger comes to a near stop, we reevaluate the spin direction according to the horizontal component of velocity.
- **Off**: A control condition in which the wheel does not spin at all. This is equivalent to $\alpha = 0$ in the previous study.
- **Shear 1D**: As the finger moves horizontally, the wheel turns slightly, causing skin deformation proportional to the horizontal velocity. Moving vertically causes no change to the wheel.
- **Shear 2D**: Similar to Shear 1D, except it also applies skin deformation when the finger moves vertically. After a sudden change in direction or when the finger comes to a near stop, the deformation direction is reset according to the horizontal component of velocity.

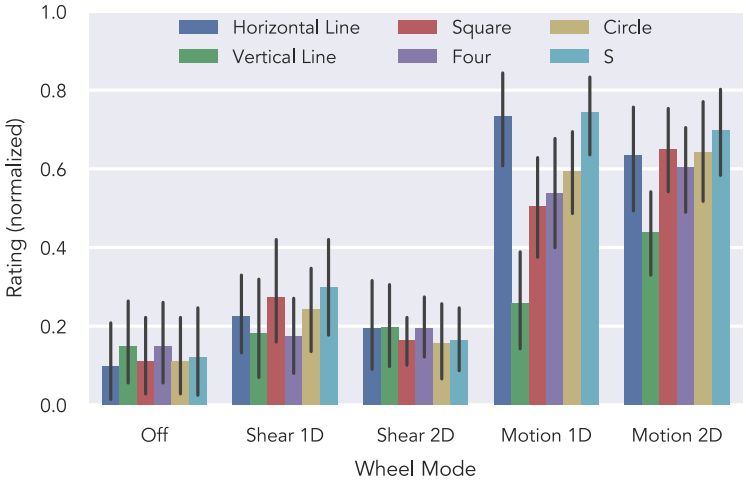

Figure 6.11: The results from the second study showing mean realism ratings across participants as a function of the wheel rendering mode and path drawn. The error bars indicate a 95% confidence interval.

As in the first study, we then asked users to rate "*How closely did the haptic rendering match your visual impression of the scene?*" on a 5-point Likert scale (1-not at all realistic, 5-highly realistic). We hypothesized that users would find the Motion conditions more realistic than the Shear or Off conditions (**H3**) and the Motion 2D condition would be most realistic (**H4**).

Results

The mean ratings from all participants are summarized in Figure 6.11. Mann-Whitney U tests show that participants perceived the Motion conditions ($n = 288$, median $= 0.67$) as more realistic than the Shear conditions ($n = 288$, median $= 0.0, U = 67382, p < 0.001$) and the Off condition ($n = 144$, median $= 0.0$, $U = 35494$, $p < 0.001$). Participants also perceived the Shear conditions as more realistic than the Off conditions ($U = 25107$, $p < 0.001$). This confirms **H3** and highlights the importance of rendering more than just skin deformation under the finger.

Ultimately, we did not find strong support for **H4** as no significant difference was observed between the Motion 1D ($n = 144$, median $= 0.67$) and Motion 2D ($n = 144$, median $= 0.67$) conditions in aggregate. However, by breaking down the analysis by path, we find some trends that suggest our 2D rendering technique is still effective. Participants perceived Motion 2D as more realistic than Motion 1D when tracing a vertical line (medians $= 0.33, 0.25$, $n = 24$, $U = 178.5$, $p = 0.021$) and a square path (medians $= 0.66, 0.5$, $n = 24$, $U = 208$, $p = 0.096$), paths which both have vertical components. No significant differences were observed with the other paths. While we expect no differences in the horizontal line, the other paths contain significant diagonal or curved components. In these cases, the difference between our Motion 1D and 2D rendering largely comes down to a difference in speed. For example, in the diagonal portion of the circular path, the 1D mode would render the motion at half the speed of the 2D mode, since only half of the motion lies in the horizontal direction. Since we are not highly sensitive to the magnitude of motion (as confirmed by Study 1), it is unsurprising that differences were not observed on these paths.

Figure 6.12: (left) A card table demo that highlights our ability to render different textures. The wheel used in this demo consists of two regions of soft felt, a hard plastic ridge, and a small section of paper. When the user touches an object in the scene, the appropriate texture is placed underneath the fingertip. (center) A painting and sculpting demo that highlights the ability to render shapes and sense the force applied to the wheel. The wheel used in this demo consists of a raised nub and a ridge to simulate holding tools. The user presses on the wheel to activate the tool. The model can be explored by touch. (right) A keyboard demo that highlights our ability to render edges and shapes. The wheel used for this demo consists of nine raised plastic regions with grooves in between. When a user approaches the edge of a key, the edge of a groove is placed under the finger.

6.5 Example Applications

To explore applications for Haptic Revolver, we built several scenes and corresponding haptic wheels that highlight different capabilities of the device. We invited 11 users to try out three of these demos in order to elicit qualitative feedback on our device and rendering techniques.

Card Table (Texture Rendering): The first application highlights the ability of Haptic Revolver to render different textures under the fingertip. In this scene, several playing cards and poker chips lie on a card table within a virtual casino. A user can touch and drag along the felt table surface, the playing cards, and the plastic poker chips and feel an appropriate texture beneath the finger. For this application, we designed a wheel containing *felt*, *plastic*, and *paper*. As shown in Figure 6.12 (left), two felt regions are used in order to render the transition from paper or plastic to felt in either direction. If a user presses lightly on one of the virtual objects, the finger will slide over it and feel the surface moving beneath it. If a user presses harder, the object will be dragged along with the finger and the device will render a shear force due to friction.

Painting and Sculpting (Force Sensing): Haptic Revolver can also turn passive props on the wheel into interactive objects by sensing the force on the wheel. In this scene, a user can paint and sculpt a 3D model by choosing between a *spray-paint tool*, a *finger painting tool*, and a *sculpting tool*. As shown in Figure 6.12 (center), the wheel consists of a raised plastic cylinder to simulate the top of a can of spray-paint and a narrow ridge to simulate the back of a knife. The tool and color can be selected by pointing at the desired element and clicking with the thumb button. When a tool is selected, the appropriate haptic element is positioned under the finger and left there until a new tool is selected. To use the tool, a user simply presses down on the haptic element beneath the fingertip. The device detects this added force on the wheel and activates the tool. When finger-painting, a smooth surface of the wheel is positioned under the finger and

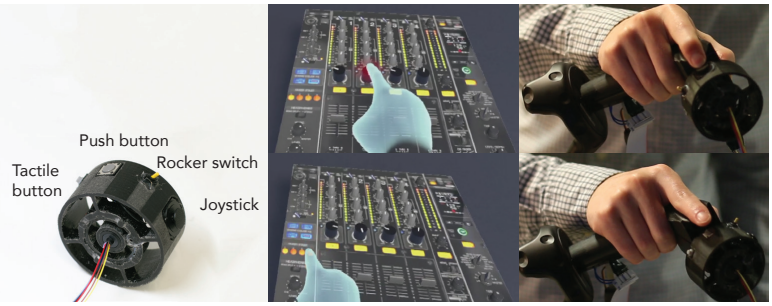

Figure 6.13: A demo with a DJ mixer board that highlights our ability to put interactive elements on the wheel. The wheel in this demo consists of several physical UI elements wired up to the device. When a user touches a virtual UI element, not only do they feel the shape of a similar physical element, but they can physical interact with the widget.

it spins back and forth during use to render shear forces and motion.

Keyboard (Shape Rendering): In this application, we highlight the ability of Haptic Revolver to render custom shapes under the fingertip and improve the experience of using virtual buttons with a purely passive wheel. In this scene, an on-screen keyboard allows a user to enter text by pressing on the virtual key with their finger. As shown in Figure 6.12 (right) the wheel for this scene consists of nine raised plastic ridges, each approximately the size of a standard key on a keyboard. Each raised "key" is separated by a small indentation to simulate the gap between keys. When the user presses on a virtual key, the rendering engine ensures the edges for the physical ridge align with the edges of the virtual key (see Figure 6.12, right). The gap can be felt by touching between two virtual keys (see Figure 6.12, bottom right). In fact, a user can lightly brush along an entire row of the keyboard and feel a rapid succession of bumps, much like one would feel on a real keyboard.

DJ Mixer (Active Wheels): While the previous applications have highlighted the capabilities of Haptic Revolver using only passive wheels, additional functionality can be added through the use of active electronic elements. In this DJ mixer application, we haptically render a virtual DJ mixer using a wheel with *active buttons*, a *rocker switch*, and a *low profile two-axis joystick*. Each widget on the mixer board is linked to a physical widget on the device. Buttons and switches have a direct mapping on the device. A joystick on the wheel metaphorically maps to virtual knobs and dials. Pushing the joystick in a particular direction rotates the knob to the same direction. Sliders are rendered using a switch on the wheel, though a simple passive object would suffice as well. When a user touches the thumb of the slider, the tactile switch is positioned under the finger. Much like the dragging behavior in the card table demo, moving the slider causes the haptic element under the finger to tug against the skin, rendering frictional shear forces. When the slider reaches its extreme point, the haptic element begins to slide off the finger.

6.5.1 User Feedback

To better understand the performance of our device, we invited an additional 11 users (10 male, 1 female) from our institution who had not tried the device before to provide feedback on its use. We sought to understand how our Haptic Revolver device compares to standard vibrotactile notification. During the

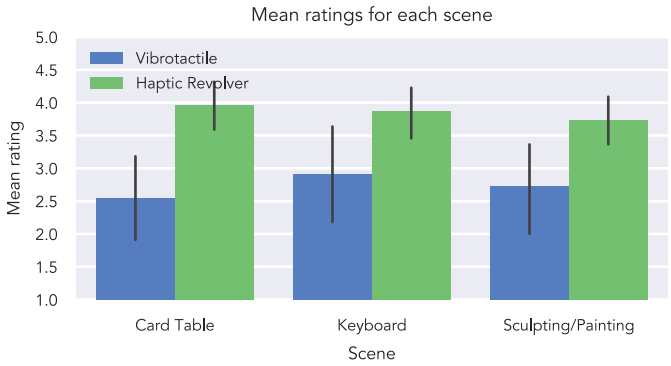

Figure 6.14: Quantitative results of our feedback elicitation study showing mean realism ratings across participants for our Haptic Revolver device and vibrotactile notification. The error bars indicate a 95% confidence interval. All differences are significant at the 0.05 significance level

study, participants tried three of our example applications: the card table scene, the keyboard scene, and the painting and sculpting scene. For simplicity and timing, we omitted the fourth DJ mixer scene. Upon arriving, users were given an introduction to the device and head-mounted display, and allowed to become accustomed to our device in a simple tutorial scene. Over the next thirty minutes, participants tried the three scenes using both our device, with the appropriate wheel for each scene, and a standard HTC VIVE controller. The VIVE controller vibrated upon contact with a virtual surface.

To elicit reactions to our device, participants explored a scene through a guided walkthrough and then provided feedback through a semi-structured interview about their experience. Questions focused on the haptic realism of various aspects of the scene, preferences related to the rendering of both devices, and usability aspects of our device. To enable a more quantitative comparison between devices, we also asked users to rate the haptic rendering ("How well did the haptic rendering match your visual impression of the scene") of each device on a 5-point Likert scale after each scene. Participants spent 3-5 minutes exploring each combination of scene and controller. Participants experienced both controllers within a scene before moving on to the next scene. We randomized the presentation order of both the scenes and the devices. Participants were compensated with an $8 meal coupon for their time.

Results

Participants were generally excited about our device and appreciated that it could render more than just vibrations. Many participants remarked that while using the Haptic Revolver, they felt like they were actually touching the surface. In the card table scene, P10 remarked, "It actually felt like I was moving my finger along a felt table". P3 noted that when touching a surface, our device responds based on how they move their hand, which felt much better than the vibrotactile controller. While using the vibrotactile notification, many users remarked that it was not the sensation they were expecting. Some users noted that while it did not feel realistic, they still appreciated the vibration feedback as an indication of touch. Users also rated our device more realistic than vibrotactile notification with consistent median ratings of 4 for our device and 3 for vibrotactile ($n = 11$) in each scene. Wilcoxon signed-rank tests between the

realism responses showed that these differences were significant for the card table ($T = 0$, $p = 0.003$), keyboard ($T = 6$, $p = 0.016$), and sculpting scenes ($T = 0$, $p = 0.003$).

This study also revealed opportunities to improve future iterations of the device. Several participants commented on the noise made by the motors. This can be improved in future iterations by using higher quality brushless motors. Some users found our reversal technique for avoiding sections of the wheel distracting, particularly when moving slowly. For example, P4 did not realize the reversal technique was intentional and commented "the motion didn't always match up with the motion of the finger". In our testing, a user's sensitivity to the reversal can be significantly reduced by ensuring the finger is pressed down against the groove on the controller, minimizing the horizontal sway during the reversal. Adding an elastic finger strap to the device may help ensure the finger rests in the ideal place. In the keyboard demo, many participants attempted to explore the sides of the keyboard, which were visually rendered as a smooth surface. Participants were surprised when they could feel edges while touching this surface. For generality, wheel designs should include a larger smooth region that can be utilized as a general purpose touch surface.

A few unexpected observations arose from this study. First, we observed differences between users with VR experience and novice or first-time VR users (seven participants had used a VR system for less than an hour). The attention of novice users was largely consumed by the novelty of the VR experience and visual aspects of the scene. With these users, it was more difficult to elicit feedback related to the haptic rendering.

Interestingly, we also observed that users started to comment on more nuanced haptic features while using our device. For instance, P4 noted that for the poker chip in the card table scene, they could feel the texture, but it was not exactly like a poker chip. As another example, P7 said "It doesn't feel exactly like spray-paint, but this is cool. I have a nice sense that I'm holding it." We suspect that because our device rendered textures and shapes in an attempt to match reality, users had higher expectations and noticed subtle imperfections, similar to the uncanny valley effect observed in 3D animation. While vibrotactile notification provided touch feedback, it was clearly not realistic and users did not expect it to be. This raises interesting questions about what level of detail is appropriate in haptic rendering for VR.

Finally, we observed that not all users preferred greater realism in the experience. For example, P8 initially felt shocked when they could feel our device rendering surface textures and remarked, "It was hard to get used to it touching my finger. It felt more real, but that was a bad thing. If I had more time I could've gotten used to that". This suggests that while it is important to explore methods of accurately rendering haptic feedback, realism may not be the only design goal.

6.6 Discussion, Limitations, and Future Work

Haptic Revolver goes beyond vibrotactile notification to explore what rendering multiple haptic sensations can add to the VR experience. While a simple haptic wheel can render touch contact and motion under the finger, the use of interchangeable haptic wheels allows applications to design custom haptic experiences. We envision some haptic wheels to be general purpose, while others may be tailored to provide highly realistic experiences for certain applications. When an end user purchases a new application that would benefit from a customized haptic experience, it could come with its own haptic wheel. Future work could

explore actuating the wheel axis forward and backward to automatically switch between multiple wheels installed on the device.

Our ability to place electronic components on the wheel significantly broadens the design space of haptic wheels. We chose to focus on interactive elements such as buttons and switches because these are commonly found in virtual environments. Our choice of physical widgets supports a wide range of virtual elements such as those found in a airplane cockpit or a car dashboard. However, electronic elements placed on the wheel are not limited to these physical controls. For example, a Peltier element on the wheel could enable temperature feedback. Adjustable mechanical components could enable additional flexibility by dynamically changing the wheel in response to the virtual environment.

We add additional interactivity to haptic wheels by sensing the force the user applies. In one of our example applications, we use this as a binary indicator of pressure to activate the spray-paint. Further interactivity could be added by using the continuous pressure signal. For example, a user might touch with varying pressure to control the spread of the paint stream.

In our keyboard demo, we improve the realism of the scene by adding ridges so that the user can feel the edges and shape of the keys. Users enjoyed this and remarked on its realism, but it is currently unclear whether this can improve performance on the keyboard in any way. Exploring how physical haptic feedback impacts task performance is an interesting avenue for future work.

Lastly, though we made an effort to design our device for users with varying hand sizes, individual differences cause subtle changes in how the fingertip rests on the wheel. This results in a different resting height for the wheel which, for some users, was manifest as physical contact before virtual contact was made. Future versions could include a proximity sensor in the wheel to automatically calibrate the correct height.

6.7 Conclusion

Haptic Revolver is a general-purpose handheld VR controller that goes beyond vibrotactile stimulation to render touch contact with virtual surfaces, motion along a surface, textures, and shapes using interchangeable haptic wheels. By customizing wheels for the virtual environment, designers can use Haptic Revolver to render realistic haptic feedback on the fingertip. We demonstrated techniques to render motion along a surface in two dimensions and adapt a particular wheel for use in arbitrary scenes. We conducted two user studies to inform and validate the design of our haptic rendering techniques and a third study to elicit qualitative feedback from participants. We believe that Haptic Revolver offers high-fidelity haptic rendering with clear advantages over vibrotactile solutions and we hope others will build upon our design to continue enabling better haptic experiences for VR.

7 WEARABLE FINGER INPUT

7.1 Introduction

While handheld controllers can provide a precise platform for tracking and a rich source of haptic feedback, they are not always appropriate for mobile input, as they require the user to hold an extra device. Other wearable form-factors can provide a useful on-the-go input modality. Such devices must carefully consider the interaction design to ensure input that subtle, yet expressive enough for mobile computing contexts. In the following two chapters, I describe approaches for interaction techniques on wearable devices and wearable accessories that enable precise finger tracking.

Thumb-to-finger interaction is a promising technique that can be performed discreetly, without large hand movements. Placing the input surface on the fingers enables fine-grained control that leverages both tactile and proprioceptive feedback. Furthermore, Huang et al. have shown that thumb-to-finger interactions are both comfortable and highly accurate [61]. Unlike many input methods that demand a particular posture during use, one can subtly swipe along a finger with the thumb with the arms at rest.

Though traditional optical hand trackers excel at hand pose detection and offer augmentation-free tracking, they do not provide enough granularity to precisely detect finger touch events and positions. Gloves, however, offer a number of advantages: freedom from occlusion and lighting problems may allow more subtle use and gloves may have fewer errors in recognizing input events. Gloves are particularly well-suited for input outside of traditional desktop computing environments (e.g. on the bus) or in situations where gloves are already commonly used (e.g. outdoors in cold weather). Because of the difficulty in creating non-contact haptics, people may also wear gloves when haptic feedback, such as force feedback or vibrotactile feedback, is desired. Though gloves might not be socially-acceptable in some situations, there is no one-size-fits-all input solution for head-mounted displays, and the ability to choose from a range of input devices for the situation will help make HMDs more ubiquitous.

This chapter describes *DigiTouch*, a touch-sensitive glove that enables thumb-to-finger interaction for eyes-free input on wearable systems. DigiTouch uses thin, partially conductive fabric strips along the fingers and a conductive patch on the thumb pad (Figure 7.1). Each strip can sense the *continuous touch position* and *pressure* of the thumb as it touches the finger. This enables precise, yet subtle input through tapping, sliding, force-pressing, and two-handed chording gestures (Figure 7.2).

Unlike other data gloves [157, 79, 85, 140], which use only discrete touch regions, DigiTouch senses the continuous touch position of the thumb. This capability makes it reconfigurable; allowing it to be used for various tasks like target selection, slider control, and text entry. Depending on application requirements, different widgets of varying size can be mapped to different regions of a particular finger. Though others have demonstrated fabric-based touch interfaces that sense continuous input, such systems either require multiple layers of fabric [54] that hinder tactile feedback, or use sensor arrays [144] making them bulky and complex. It is also unclear how well these systems operate when bent or stretched, as doing so can

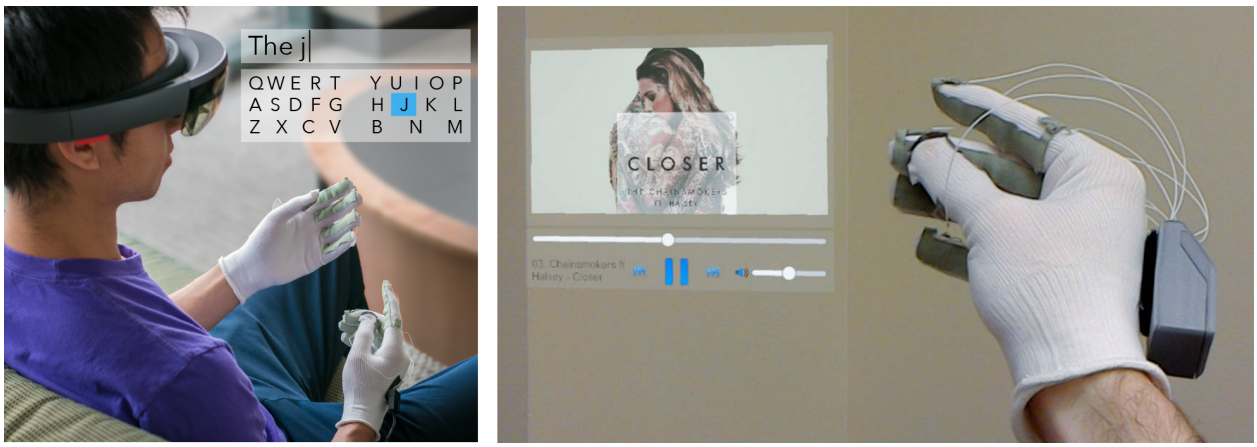

Figure 7.1: DigiTouch is a touch-sensitive glove that explores the use of thumb-to-finger interaction for input and text entry. (left) Such an input technique is well-suited for use with virtual and augmented reality. (right) Controlling a HoloLens application using DigiTouch.

change the electrical properties of fabrics. To overcome these challenges, we present a new technique for continuous sensing on fabric that uses only a single layer of fabric and a two-wire interface on each finger. DigiTouch accounts for the variable resistance as the fingers bend using current monitoring and time-multiplexed sensing.

DigiTouch is a general-purpose input device for AR/VR systems and can be used for different applications (Figure 7.2). For example, a user can dial a number using a ten-digit numeric keypad, move a virtual object by sliding along a finger, or control an application using any combination of buttons and sliders along the fingers. However, in evaluating DigiTouch, we decided to place special emphasis on text input using a split-QWERTY keyboard. The reason is three-fold: (1) Text entry is a challenging task in today's AR/VR systems and is a barrier to enabling more productive use-cases. (2) There is limited quantitative data on the evaluation of such wearable input systems for AR/VR applications, and text entry provides a well-established set of quantitative measures that helps in formalizing the system's performance. (3) The high density of keys using a full QWERTY keyboard in a fixed 2-dimensional space makes text entry a challenging task for the user, and a rigorous test for the usability and practicality of DigiTouch. We directly map a split-QWERTY keyboard layout to a user's fingers, as shown in Figure 7.2 (right). This closely resembles the two-thumb typing posture on a smartphone. From a longitudinal study with ten participants, we found that the participants quickly learned how to use DigiTouch for entering text. Their mean typing speed increased from 7.0 wpm (words per minute) to 16.0 wpm in 10 twenty-minute sessions. The participants also achieved a mean uncorrected error rate of 0.85% on the last session.

The main contributions of DigiTouch are:

1. A reconfigurable touch-sensitive glove that senses continuous touch position and pressure, enabling thumb-to-finger interactions for wearable computing.

2. A text entry system using thumb-to-finger interactions based on a split-QWERTY keyboard.

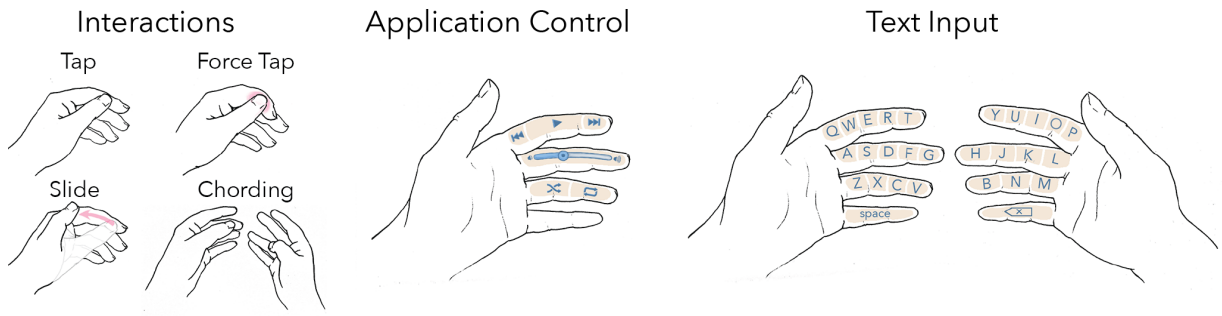

Figure 7.2: (left) *DigiTouch* enables tapping, force tapping (tapping with pressure), swiping, and chording (two handed tapping) gestures. (center) Because DigiTouch senses continuous touch position, it enables an arbitrary configuration of input widgets, which can be customized for a particular application. (right) DigiTouch also enables text input by mapping a full QWERTY keyboard to the hands and reserving the pinky fingers for the space and backspace keys.

3. A quantitative evaluation of the text entry capabilities of DigiTouch using a ten-session study with 10 participants.

7.2 DigiTouch System

In this section, we discuss the design of interactions enabled by DigiTouch, followed by the hardware and software implementation of the system.

7.2.1 DigiTouch Interactions

DigiTouch interactions are based on thumb-to-finger touches. It is a continuous-input mechanism that allows the thumb to manipulate virtual widgets placed along the fingers. This is enabled by the unique flexibility of the thumb, whose adduction and flexion ability enables it to comfortably touch the other four fingers. It provides both tactile and proprioceptive cues to the user, enabling eyes-free accurate touch locations. Also, as each finger is separated into three phalanges, the joints between these segments serve as natural reference points for the user. *DigitSpace* [61] explored the comfort and accuracy of thumb-to-finger touches. They found that users can comfortably and accurately discriminate between up to five buttons on most fingers, in an eyes-free manner. We leverage these findings to inform our interaction design.

We explore interactions that rely on the following touch gestures: *tap*, *force tap*, *slide*, and *chording* (touching two fingers of opposite hands simultaneously). These simple gestures act as building blocks and enable a wide range of functionality, including typical buttons, pressure-sensitive buttons, sliders, chording input, and even a full keyboard for text input. DigiTouch's continuous sensing capability ensures that the input is reconfigurable and various input controls (such as buttons, sliders, etc.) can be mapped to finger positions according to the needs of an application. For example, a music player may use only a few buttons

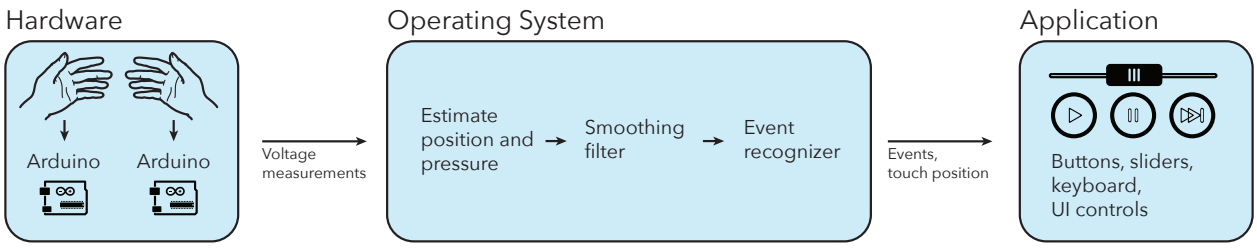

Figure 7.3: Overview of DigiTouch tracking system. The hardware layer measures voltages from the gloves and transmits them to a computer for processing. The operating system layer on the computer estimates the touch position and pressure, smooths the signals, and recognizes the event. Finally, applications can implement user interfaces on top of this layer.

per finger on a single hand (Figure 7.2, (center)), whereas a 3D modeling application may use fine-grained controls with up to five buttons and/or sliders per finger.

7.2.2 *DigiTouch Glove Design*

DigiTouch is implemented as a glove-based system since gloves provide the most reliable way to detect contact between the thumb and finger. Other tracking techniques may be able to provide more accurate hand pose estimation or positional tracking, but are less reliable at detecting actual contact events. Our system consists of two gloves with a touch strip along the length of each of the four fingers and a conductive patch on the pad of the thumb. Each glove is powered by an Arduino Trinket Pro which streams data to a computer over a wired serial connection. Though not used in the evaluation, we also implemented a wireless version of the gloves that streams data using the GZLL wireless protocol. Software on the computer receives, processes, and filters the data to determine touch position and pressure on each hand. Based on this information, it triggers events that contain information about the touch position and pressure and describe the hand's state, such as *OnThumbDown* and *OnThumbMove*. It then sets up a web socket for any web-based client to receive the touch events. We implemented clients on a HoloLens as well as a standard web browser. Figure 7.3 shows a high-level overview of the DigiTouch system.

While constructing the DigiTouch glove, our main design considerations were: the glove fabric should be thin and elastic, ensuring good contact with the skin and preserving tactile feedback, and the glove should be comfortable, so that it can be worn for extended periods of time. DigiTouch (Figure 7.4) consists of three main components: (1) the thin elastic nylon glove, (2) partially conductive fabric strips that act as linear touch sensors on four fingers, and (3) conductive fabric to make a thumb patch. We chose to use conductive fabric over other conductive materials because it is flexible and maintains its conductivity over time. The partially conductive fabric strips are made from a polyester/cotton blend with small stainless steel fibers to make it partially conductive[1]. The thumb patch is made from a cotton woven with stainless

[1]Staticot, from Less EMF

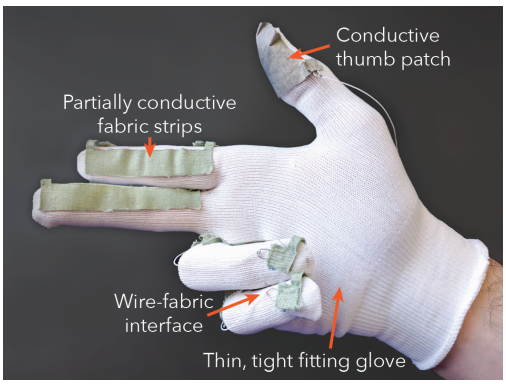

Figure 7.4: DigiTouch consists of a conductive thumb patch that makes contact with one of the partially conductive fabric strips along the fingers.

steel thread[2]. While we attach commercially available fabrics to an existing glove, we note that recent advances in digital textiles [126] will enable tighter coupling between the glove and the electronics. Future designs could have the conductive strips woven into the glove itself.

With this configuration, we ensure that the user can touch with any portion of the thumb, but the contact point on the finger strip determines the touch position. This is important because depending on anatomical differences, which finger is being touched, and the orientation of the thumb, a different part of the thumb will make contact with the finger. Thin wires are attached to each end of the finger strip and one end of the thumb patch using conductive silver epoxy (Figure 7.4). This provides a reliable wire-fabric interface that does not change with movement.

7.2.3 *DigiTouch Sensing*

The fabric strip on the fingers has a resistance that is approximately uniform across the length of the strip. Each strip has a total resistance of approximately $500\,\Omega$. We model each finger as a potentiometer, and the thumb acts as a wiper that slides along the resistive fabric. Figure 7.5 (left) shows a simplified schematic of the glove with one finger and one thumb. When the thumb makes contact with a portion of the finger: R_1 represents the resistance of the partially conductive fabric between the knuckle and thumb, and R_2 represents the resistance of the partially conductive fabric between the thumb and fingertip. A voltage, V_{stim} (3.3 V, nominal), is applied to the thumb patch and $R_{current}$ is a constant physical resistor that limits current flow and allows us to treat the circuit as a voltage divider. By measuring the voltage at the base of the finger (V_{finger}) and connecting the fingertip to ground, we can estimate R_2 according to Equation 7.1.

$$V_{finger} = V_{stim} \frac{R_2}{R_{current} + R_2} \tag{7.1}$$

[2]HertzCloth, from Less EMF

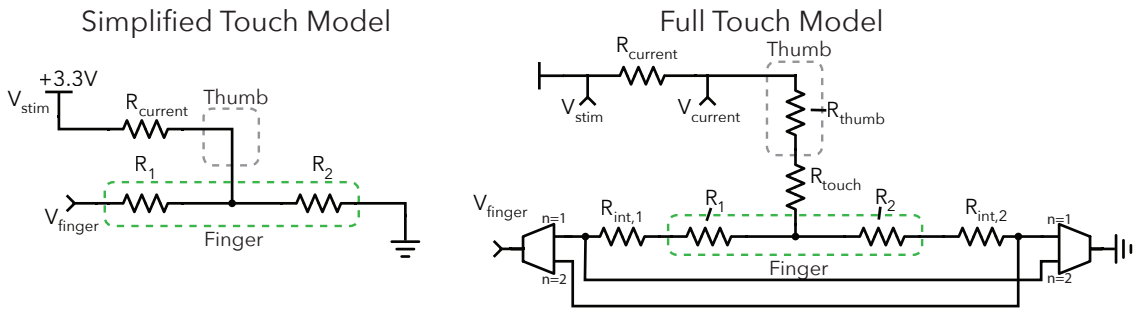

Figure 7.5: (left) Simplified circuit model of the thumb and finger circuit. Current flows through $R_{current}$, through the thumb, and makes contact with the finger between R_1 and R_2. (right) Full circuit model showing all voltage measurements and multiplexed input. Each end of the finger is alternately ($n = 1, 2$) grounded while the voltage at the other end is measured. Since touching with increased pressure reduces the resistance at the fabric-fabric interface, estimating R_{touch} gives an estimate of touch pressure.

Assuming that the overall resistance of the fabric ($R_T = R_1 + R_2$) is constant, we can estimate the touch location, x according Equation 7.2.

$$x = \frac{R_T - R_2}{R_T} \tag{7.2}$$

Unfortunately, this simple design presents several challenges. First, the overall resistance of the cloth (R_T) changes significantly with finger bending/stretching, and over time due to environmental factors. Second, since the conductive thumb pad contacts the finger over an area, the overall resistance between the two ends of the finger strips is reduced. Third, this model assumes the user makes perfect contact between the thumb and the finger; however, inconsistent touch pressure can result in additional resistance.

We took several steps to address these challenges, thus estimating touch pressure and accounting for dynamic changes in fabric resistance. Figure 7.5 (right) shows the full circuit design with one finger and one thumb. In this model, we account for additional resistances at each of the wire-fabric interfaces ($R_{thumb}, R_{int,1}, R_{int,2}$) and the fabric-fabric interface resistance at the touch location (R_{touch}). We also use an eight-channel digital multiplexer and make two sets of measurements to account for the variable resistance in R_T, due to bending/stretching of fingers. In Figure 7.5 (right), only two of the eight channels are depicted. The remaining channels are connected to the other fingers in a similar manner. When the user's thumb is not touching a finger, the microcontroller toggles between the eight multiplexer channels through three digital logic lines. By toggling these, the microcontroller rapidly switches the ground and analog-to-digital converter (ADC) connections between the tops and bottoms of the four fingers, such that when one side of a finger strip is grounded, the other is connected to the ADC. As soon as a touch is detected, it switches to a focused-mode where it rapidly toggles only between the top and bottom of the finger being touched to increase the data rate. The non-grounded end is measured with an ADC (V_{finger}). We also measure current flowing through the thumb by measuring the voltage drop across the current limiting resistor.

To summarize, for each finger, we measure the following: $V_{stim,1}$, $V_{current,1}$, and $V_{finger,1}$, when the inside

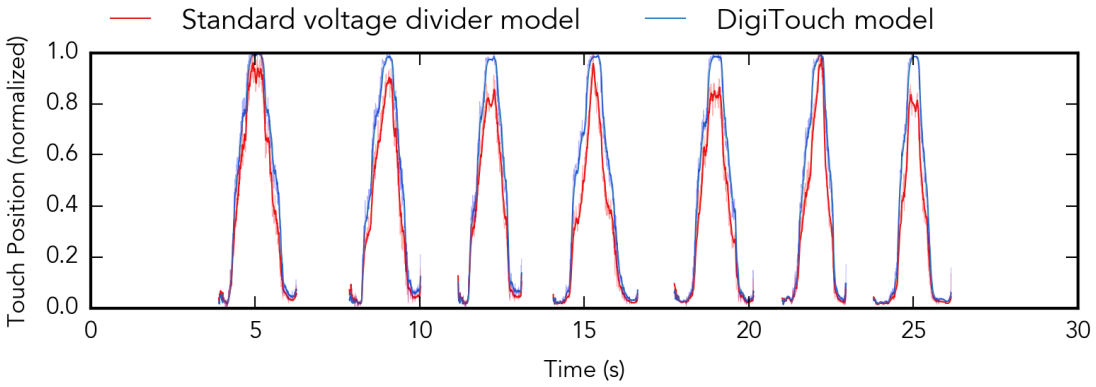

Figure 7.6: A series of swipes starting at the base of the finger (position = 0), moving to the tip of the finger (position = 1), and returning to the base (position = 0). Using the traditional voltage divider model (red) with ground at the fingertip, the total resistance drifts over time. In DigiTouch we measure the finger resistance from both the fingertip and the finger base, accounting for any resistance drifts over time.

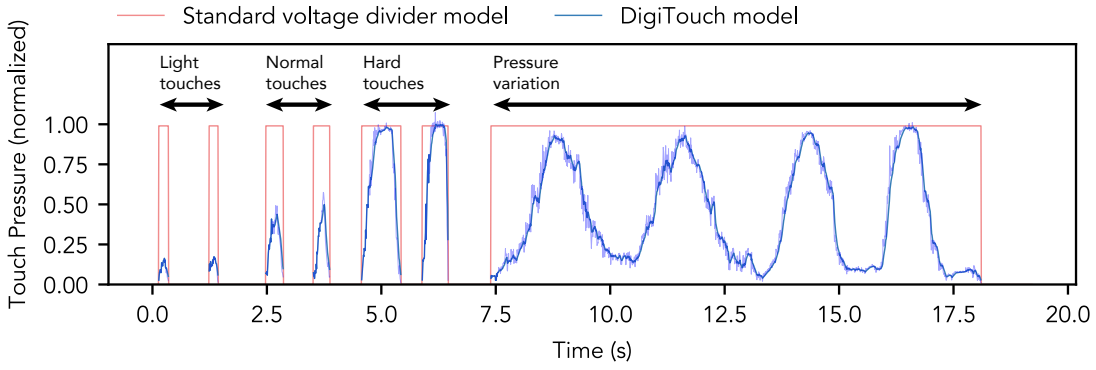

Figure 7.7: DigiTouch pressure signal when a user performs two light touches, two medium touches, and two hard touches. Finally the user maintained contact and varied the contact pressure. The DigiTouch model enables force-sensitive interaction, while a standard voltage divider model would only detect a binary touch event.

of the finger is grounded and $V_{stim,2}$, $V_{current,2}$, $V_{finger,2}$ when the outside of the finger is grounded. R_{thumb}, $R_{int,1}$, and $R_{int,2}$ are constants that can be set with a one-time calibration.

We first estimate the current flowing through the thumb for each state ($n = 1$, $n = 2$) according to Equation 7.3. We then compute the resistance of each section of the finger strip (R_1, R_2) according to Equation 7.4. We compute touch position as before (Equation 7.2).

$$I_n = \frac{V_{stim,n} - V_{current,n}}{R_{current}} \tag{7.3}$$

$$R_n = \frac{V_{finger,n}}{I_n} - R_{int,n} \tag{7.4}$$

Figure 7.6 shows an example of how our model accounts for the varying resistance of the fabric strips during a typical swipe gesture. In a typical voltage divider model, when the electrical properties of the fabric change over time due to mechanical changes in the fabric or sweating of the user, it causes the total resistance to drift. With our time-multiplexed approach, we no longer depend on knowing the total resistance of the fabric strip and we can more consistently estimate the touch position.

Finally, we estimate the pressure between the thumb and finger by computing R_{touch} according to Equation 7.5. We model pressure as the resistance between the fabric-fabric interface. As the user presses harder, both the contact area and the conductivity of the contact area increase. We compute the pressure estimate twice, using each set of measurements ($n = 1, 2$) independently and average them together. The pressure output is computed according to Equation 7.6, where k is a constant set to the minimum possible touch resistance.

$$R_{touch,n} = \frac{V_{stim,n}}{I_n} - R_{current} - R_n - R_{extra,n} - R_{thumb} \tag{7.5}$$

$$Pressure = \frac{k}{\max(k, \frac{R_{touch,1}+R_{touch,2}}{2})} \tag{7.6}$$

Figure 7.7 shows an example of the pressure signal measured by DigiTouch. In this example, the user performs a series of light, normal, and hard touches on the index finger, followed by ten seconds of continuous contact while varying the pressure. This illustrates the ability to enable force-sensitive interactions, such as those enabled by the 3D touch features on iOS devices.

7.2.4 *DigiTouch Signal Processing*

Raw data is processed by software on a computer that streams data from the glove over two serial port connections. Though this implementation uses a wired setup, it could be easily made wireless using Bluetooth or a similar protocol. Each message frame contains information about which finger was touched and the six voltage measurements describing the touch state. During a touch, data is sampled and processed at 160 Hz. At all other times, the data stream is monitored at 70 Hz. For each set of measurements, the software computes the contact position and touch pressure, and dispatches relevant events that took place since the last measurements. The event model includes *OnThumbUp*/*Down* events, an *OnThumbMove* event, and *OnBeginForce*/*OnEndForce* events. Force-based events are triggered when the user's touch pressure exceeds a threshold.

Because of inconsistent touch contact and movement of the fingers, the raw signal can be noisy. We therefore pass all six voltages signals through an exponential smoothing filter ($\alpha = 0.2$) to smooth them before any computation is performed. At the start of a touch event, these filters snap to the first detected set of measurements, to minimize lag. These smoothed values are then used to compute the touch position and pressure, as previously described. Since the signal changes rapidly when the user first touches their thumb to a finger, we implemented a second stage exponential smoothing filter for the position and pressure signals. This filter uses a dynamic smoothing factor that starts high ($\alpha = 1$, no smoothing) when the user first touches and decreases throughout the touch event (α drops 0.2 per frame, down to a limit of $\alpha = 0.2$). These parameters are reset on every touch. This helps minimize the effects of drift when the user first makes contact and makes the system feel more responsive.

We use a finite state machine to recognize events. The state machine keeps track of the current conceptual state of the user's interaction with the glove ("not touching", "touching but holding still", or "moving", in addition to how hard the user is pressing). Event transitions are determined by the changes in the position (for *OnThumbMove*) and pressure (all other events) signals. An event is fired every time the machine transitions, along with important details associated with the event (such as position for an *OnThumbUp* event, or starting and ending positions for an *OnThumbMove* event). The sequence of events that are triggered for several user actions are:

- Tap: *OnThumbDown* $\rightarrow$ *OnThumbUp*;
- Force Tap: *OnThumbDown* $\rightarrow$ *OnBeginForce* $\rightarrow$ *OnEndForce* $\rightarrow$ *OnThumbUp*;
- Slide: *OnThumbDown* $\rightarrow$ *OnThumbMove* $\rightarrow$ *OnThumbUp*.

Finally, the detected position, pressure, and events are written to a WebSocket, ready for consumption by a web-based user interface widget.

7.3 Text Entry Keyboard Design

We chose text entry using a full QWERTY keyboard to study the performance of DigiTouch in a real-world task. Text entry is a complex task requiring more than 26 buttons, which is hard to achieve using most input modalities. Most devices designed for wearable input rely on chording [90, 85, 140] due to a limited number of buttons. This leads to an increased cognitive load as users learn to use the system. On DigiTouch, we created 28 keys (26 letters + *space* + *backspace*) accessible by at most one tap on the finger using the thumb. The keyboard layout is illustrated in Figure 7.2 (right).

We chose the standard QWERTY keyboard as it is most familiar to people. In a QWERTY layout, there are at most ten letters in each row. By splitting the QWERTY keyboard in two halves, we ensure there are never more than five buttons assigned to each finger. In this layout, each thumb acts as a *stylus* and effectively operates on one half of the keyboard. Thus, it closely resembles two-thumb typing on a smartphone or tablet. In fact, many users are already familiar with a split-keyboard layout, since it is a feature on both Android and iOS for tablets. Because of this familiarity, we expect a smaller learning curve compared to chording-based text entry systems.

Huang et al. [61] estimated the maximum number of buttons that can be placed on each finger before the target selection performance becomes unusable. They reported that users could trigger five buttons per

index and middle finger, four buttons per ring finger, and two buttons per pinky finger, with high accuracy. Our keyboard layout falls within their suggested guidelines.

The space and backspace keys are the two most commonly used keys on a keyboard [183]. To avoid frustration for users, they must not be confused with other keys. We decided to dedicate the left and right pinky fingers to the Space and Backspace keys, respectively (Figure 7.2, right). In the future, one can imagine adding control or punctuation keys to the outsides of pinky fingers.

Though tactile and proprioceptive feedback helps in target selection, users might not always press the correct target initially, particularly on fingers with four or five buttons. Because DigiTouch can track the continuous position of the thumb sliding along the finger, we mimic the interaction on modern smartphone keyboards and allow the user to slide their thumb along the fingers to select the correct key. On a thumb-down event, a letter gets highlighted. The user is free to slide the thumb on the finger to switch to adjacent letters. When the user's thumb is over the letter that they wish to press, a thumb-up event triggers letter selection.

7.4 Text Entry Evaluation

7.4.1 Study Design

The goals of our controlled evaluation was to observe: (1) whether the participants were able to type using the thumb-to-finger interactions enabled by DigiTouch; (2) how their performance varied over time and with practice; and (3) whether personalized keyboard model impacts performance.

Participants

Ten participants (7 male, 3 female), with a mean age of 23.1 years (18 - 27 years) participated in the study. Eight were right-handed (two left-handed), and all self-rated as expert touch screen smartphone users. Each participant was compensated $5 per twenty-minute session.

Apparatus

Participants interacted with our custom text entry software (Figure 7.8) running in a web browser (Google Chrome) on a Windows 7 desktop computer. Participants sat on an adjustable reclining chair and were asked to keep their hands in any comfortable position. In order to evaluate eyes-free input, they were simply asked to position their hands such that they are unable to see their hands while typing (by keeping their hands in their lap or by putting their hands under the table). The software logged all of the users' touch events.

Procedure

The procedure was designed to fit in ten 20-minutes sessions spread over a period of 15 days, with consecutive sessions separated by at least six hours and no more than two days. However, due to scheduling

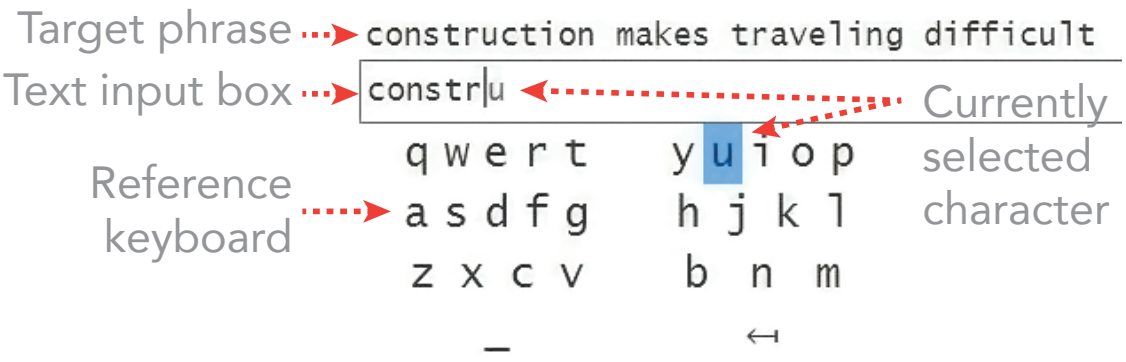

Figure 7.8: Custom web application for the text entry user study. Participants were presented with a target phrase and instructed to type it as quickly and accurately as possible. Participants could see their input beneath the target phrase. A reference keyboard showed the key placement on the fingers and highlighted the currently selected key.

conflicts, some participants exceeded this two-day limit. For each participant, the first session began with an introduction to DigiTouch, the text entry task and the experiment software. At the beginning of each session participants were asked to swipe along each strip to ensure proper connection and fit. Occasionally, a wire would come loose during a session due to strain. In such events the session was paused while the connection was restored, and the current phrase was thrown out.

For each 20-minute session, participants were asked to input as many phrases as possible, similar to [26]. The phrases were randomly chosen from a published phrase set for text entry by MacKenzie and Soukoreff [93], with average phrase length of 28.61 characters. The experiment was conducted only with the lower case letters (no upper case letters, punctuation, or numbers). All the participants received the phrases in a randomized order. We asked the participants to type "as quickly and accurately as possible", and to fix errors unless those errors were made "far behind" their current point of entry. Participants were encouraged to take a short break between phrases, anytime they wished. The user interface (Figure 7.8) showed phrase to be transcribed (target phrase), a text box to receive participants' input, a timer, and a 'Pause' button at the bottom. After completion of a phrase, participants used a chording gesture by simultaneously touching both thumbs to their respective index finger to move on to the next phrase.

7.4.2 *Text Entry Results*

All ten participants completed ten sessions. In total, participants entered 3686 phrases. The main measures for evaluating the performance of DigiTouch were typing speed, corrected error rate, and uncorrected error rate. We also conducted an analysis of the input stream to arrive at character level metrics that characterize performance of different regions of the glove. Unless otherwise noted, all reported means and standard deviations are computed across participants and phrases.

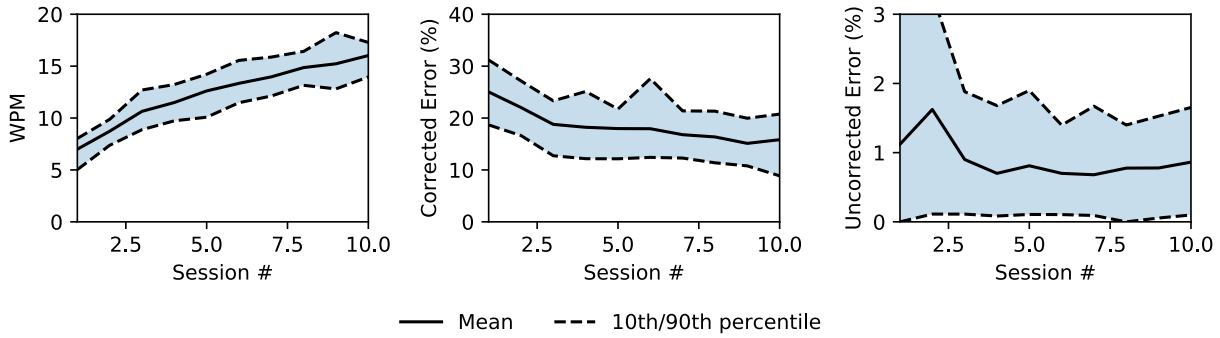

Figure 7.9: Summary of text entry study results. (left) Typing speed measured in words per minute (wpm) increased with practice. (center) Corrected error rate, a measure of errors that users eventually corrected, decreased with practice. (right) There were no clear trends in the uncorrected error rate, a measure of errors remaining in the final transcription.

Speed

Text entry speed was measured in words per minute (wpm), calculated as (characters per second) $\times \frac{60}{5}$, using the assumption that a word consists of 5 characters [92]. The average text entry speed over all sessions and across all participants was 13.0 wpm ($\sigma = 4.12$). The mean speed for the first session was 7.0 wpm ($\sigma = 2.2$), while the mean speed for session 10 was 16.0 wpm ($\sigma = 4.1$), clearly showing that performance increased with practice (Figure 7.9). The learnability curves obtained (Figure 7.9) are similar to the characteristic learnability curve for text entry system [90].

Speed at a character level was analyzed using *dwell time*, measured as the time difference between the thumb down and the following thumb up event. Dwell time for the first session, averaged over all the participants was $443\,\mathrm{ms}$ ($\sigma = 124\,\mathrm{ms}$), while for the final session, it was $301\,\mathrm{ms}$ ($\sigma = 74\,\mathrm{ms}$). This shows that time taken to input a character reduces with practice over time. Figure 7.10 shows the average dwell time for each correctly entered character during the final session. Lower dwell time for the characters situated on the extremes suggest users are more confident in pressing these characters.

Accuracy

Two metrics were used to measure text entry accuracy: (a) Corrected error rate [152] – a measure of the errors that the user corrected in the final transcription, and (b) Uncorrected error rate [152] – a measure of the errors remaining in the transcribed text that the user did not correct. A user's typing speed represents a trade-off between corrected and uncorrected errors. More corrections result in a slower typing speed, as each correction adds multiple keystrokes, i.e., backspace character, re-enter character. These metrics were computed using software developed by Wobbrock et al. [183].

The mean corrected error rate for session 1 was 25.0% ($\sigma = 11\%$), and for session 10 was 15.8% ($\sigma = 10\%$). For the last session, this means that 15.8% of all characters typed were ultimately incorrect characters that

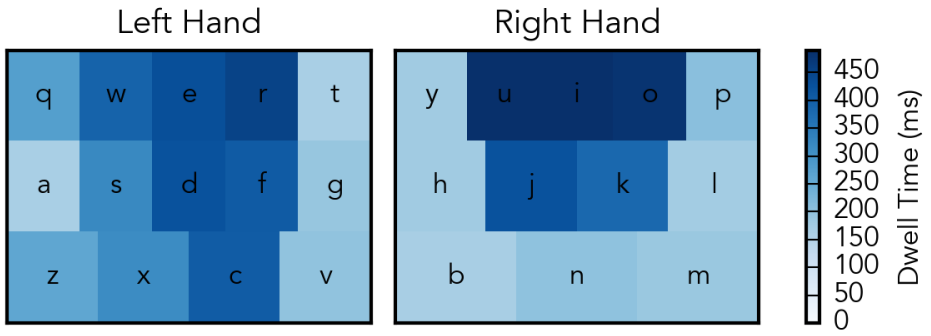

Figure 7.10: Heatmap showing mean dwell time for each correctly pressed key. The time indicates how long the user held the thumb down to eventually type each character.

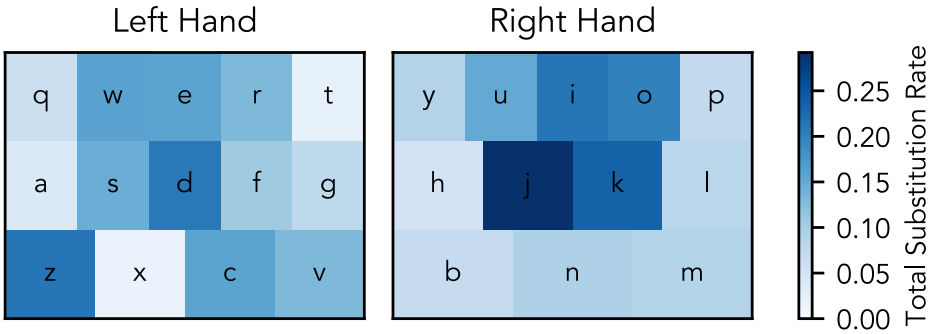

Figure 7.11: Heatmap showing total substitution rate for each key. Higher errors indicate less accuracy in attempting to type a particular character.

the user fixed. The mean uncorrected error rate across all sessions was 0.85% ($\sigma = 2.3\%$), and there was no clear trend across sessions (Figure 7.9, right). This suggests that participants' tolerance for error showed little variation with practice. Since the corrected error rate dropped and was accompanied by no distinct increase in uncorrected error rate, this suggests that the number of errors users had to correct decreased with practice (Figure 7.9, center).

Input Stream Analysis

These error rates consider participant's accuracy on a phrase level, but more nuanced results can be obtained from an analysis of the input stream. We use the approach developed by Wobbrock et al. [183] to compute all possible alignments between the input stream and the intended output. The *character-wise total substitution rate* (Figure 7.11) answers the question, "when trying for *i*, what is the probability that

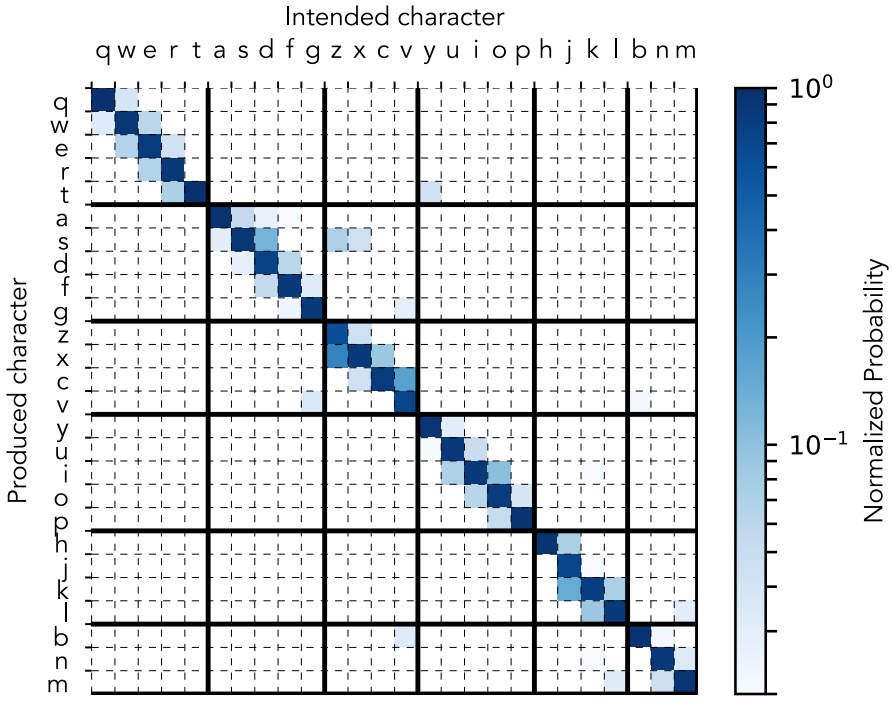

Figure 7.12: Confusion matrix showing the probability of produced vs intended characters according to an analysis of the input stream. The plot is ordered by the placement of keys along the finger so that the most often confused letters appear closer to the diagonal. The scale is logarithmic to better highlight the confusions.

participants did not get *i*?". This allows us to examine which characters were most prone to confusion. In general, participants were more accurate on characters situated on the extremes of the fingers, and characters situated on fingers with fewer buttons. The highest substitution rate was seen for the characters 'j' and 'z'. This may be due to the fact that they occurred very few times in the target phrase ('j' 38 times, 'z' 8 times, compared to 'e' 1480 times), thus making these errors less significant.

A confusion matrix represents the frequency of character-level errors [91]. Figure 7.12 shows the total number of times when an intended character was transcribed with a (correct/incorrect) produced character. The most prevalent mistakes were: transcribing 'i' instead of 'o', 'k' instead of 'j', etc. Since most of the errors in the confusion matrix are adjacent letter confusion, an auto-correct can dramatically boost the text entry performance.

7.5 Pressure Evaluation

While the text entry study evaluates the touch sensing capabilities of DigiTouch, it does not exercise the pressure sensing functionality. To evaluate the pressure sensing capabilities of DigiTouch, we conducted

a short user study with 10 participants (6 male, 4 female) with a mean age of 21.9 years (18 - 25). The experiment was split into two phases and took approximately 10 minutes. In the first phase, participants were asked to touch the thumb to the index finger with either *light* or *hard* pressure. In the second phase, a third level was added (*light*, *medium*, and *hard*). In both phases, the order was randomly presented and randomly switched between the left and right hand to minimize fatigue. The user was presented with the type of touch to perform and after a 1 second delay to prevent users from rushing, a tone was played to indicate that the user could perform the touch. When an up event was detected, the pressure recorded was the maximum detected pressure during the touch event.

The pressure boundaries between levels were set based on the results of a pilot study that was conductive with a separate set of five participants. Before each phase, participants first learned the pressure boundaries during a practice period in which they received visual feedback on the pressure. Once participants performed the tap with the correct pressure, they were allowed to move on. During the actual experiment, participants performed the same task without receiving any kind of feedback. Participants performed 5 taps per level per hand for practice and 10 taps per level per hand during the evaluation.

In total, 1000 touches were collected from this study. DigiTouch identified the correct pressure on these touches with an accuracy of 93.3% in the two-level case (N=400) and 64.0% in the three-level case. The distribution of touches for each type of touch are presented in Figure 7.13. Hard touches were more tightly distributed while there was a broader distribution of light touches. Though we assume that the user always pressed with the intended touch pressure, we note that this may not always be the case, particularly for the three-level case. Several participants reported a conceptual difficulty in distinguishing between three different pressure levels. Others reported that they could perform the three-level task, but they had to concentrate harder. Users universally reported that the two-pressure task was easy. While applications may benefit from discrete pressure-enabled input, the relatively poor observed performance with three pressure levels suggests that reliably distinguishing more than two levels is difficult.

7.6 Discussion

7.6.1 Glove Design

Head-mounted displays have the potential to enable truly ubiquitous computing. Though a universal input solution is difficult because different situations demand different capabilities, a user should be able to use the right technique for the situation. Prior attempts to add input to gloves have produced bulky systems with low adoption. Our sensing technique enables continuous input with relatively little instrumentation, compared to other glove-based devices. To achieve this style of continuous tracking with traditional techniques would require significantly more instrumentation, using multiple fabric layers or many wired contact points. Moreover, AR and VR present use cases for gloves that may overcome prior hurdles to adoption. For example, certain capabilities, such as high-fidelity haptic feedback, are likely impossible without the use of gloves. Also, in cases where precise hand tracking is required, gloves can simplify the task of pose estimation [175].

DigiTouch is particularly compelling when combined with traditional optical hand tracking techniques, which excel at pose estimation, though are not precise enough to detect thumb touch events or touch locations. With the hand pose and segmentation from a hand tracker, the virtual widget layout can be

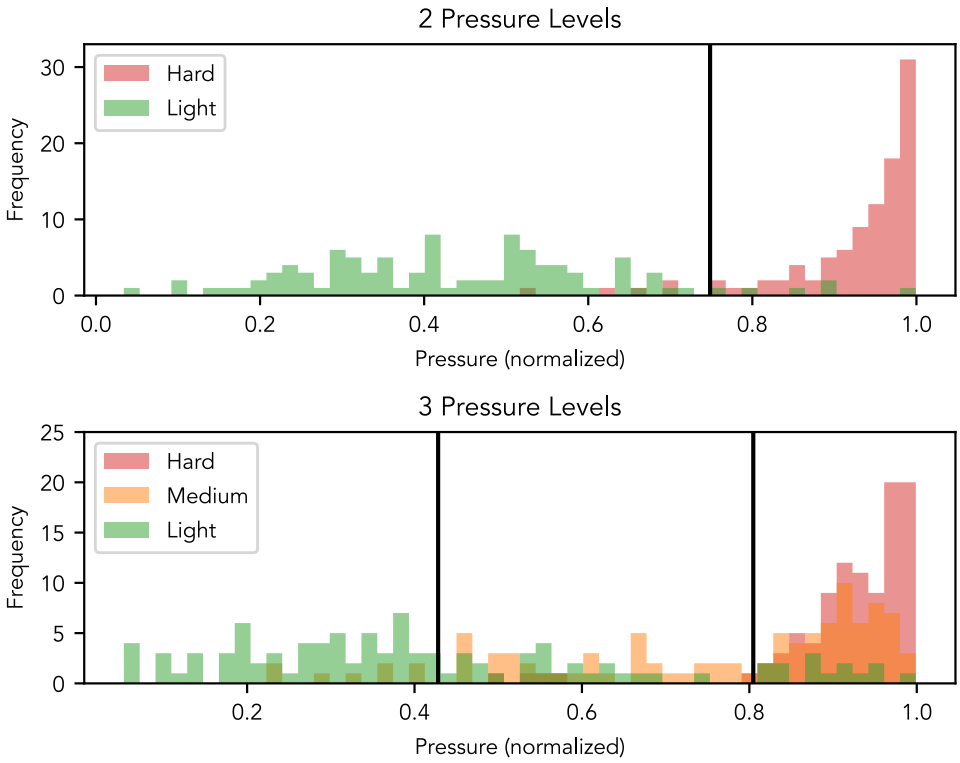

Figure 7.13: Distribution of measured touch pressures when asked to touch either light or hard (top) or light, medium, or hard (bottom). Note that the distributions are drawn partly transparent in order to view the overlapping distributions underneath. The black lines indicate the predefined boundaries between pressure levels.

superimposed directly on the user's fingers. For example, in a text entry system, a user might see the individual characters superimposed on their fingers at the correct location. This would significantly improve the learnability of such a system.

With any wearable system, there is a concern for false activations. This is particularly important with any kind of hand-based sensing, since we use our hands for so many things. These issues will arise primarily in an AR scenario. We envision that such a system can have an activation gesture that enables continuous position and pressure sensing on all fingers. With DigiTouch, a number of different activation gestures are possible. For example, users could tap on each finger sequentially or swipe along a particular finger to enable input.

One challenge we observed with thumb-to-finger input is determining the precise location of a touch point. When users touch their thumb to a finger, they make contact at an area and it can be unclear where the intended touch point is, as it depends on the target finger and the orientation of the thumb. In an initial prototype of the DigiTouch glove, we used a small thumb patch to minimize the ambiguity in the touch location. However, we found that this caused significant frustration for users since they would often miss the finger strips completely. As a result, we settled on a large patch that covers most of the thumb in order to maximize the reliability of touch detection at the expense of precision in estimating touch position. As one might expect, this also causes some issues with drifting touch locations as the user rolls and lifts the thumb from the finger at the start and end of touch events. In DigiTouch, we counter this drift by modifying filter parameters to minimize drift at the start and end of touch. In *Understanding Touch*, Holz and Baudisch explore a closely related problem with touchscreens on mobile devices [57]. With additional study, it may be possible to construct a model for intended touch location as a function of the thumb position and contact area. This may enable a dynamic correction that could significantly improve touch precision.

7.6.2 Using DigiTouch for Text Entry

With any new input system, there are many aspects to evaluate. For DigiTouch, we chose to evaluate one of the most challenging use cases that covers most of the interaction space. In our proposed keyboard design, users tapped and swiped to input a character, and used chording to advance to the next phrase. We use the ability of users to input text in this manner to claim that users would also be effective at controlling other applications with custom layouts.

The results from the text entry evaluation demonstrate that users were able to effectively type using DigiTouch. User performance in our study exceeded that of many other similar wearable text entry systems. Rosenberg et al. [140] showed a mean typing speed of 8.9 wpm after 80 minutes and 16.8 wpm after nearly 10 hours of practice. DigiTouch achieves a similar speed after only 3.3 hours of practice. The input glove designed by Hsieh et al achieved a mean typing speed of 5.4 wpm after twenty minutes of practice [59]. In the same amount of time, DigiTouch users achieved a similar typing speed of 6.5 wpm. Lyons et al evaluated the learnability of Twiddler[3], a commercial product often used for text entry on wearable systems. They found that the mean text entry speed was 19.5 wpm after 200 minutes, and increased to 47 wpm after

[3] http://twiddler.tekgear.com/

25 hours of practice [90]. Though our study design was limited to 200 minutes of practice, DigiTouch performed comparably at 16.5 wpm (vs 19.5 wpm). By looking at the typing speed progression over sessions (Figure 7.9, left), it is hard to judge improvement beyond the initial 10 sessions. However, the continuous nature of DigiTouch enables additional controls like application specific layouts and sliders, something that cannot be achieved with a Twiddler.

Most users during the study chose to hold their hands apart, with their arms resting on the chair or at their side. This suggests that users were comfortable with the mental model of the split keyboard. It also highlights an advantage of DigiTouch over other hand-based interaction techniques that requires the hands to be held up. DigiTouch can easily be used with the arms at rest and may even be appropriate for use while walking.

During the study, one participant noted that after thinking of the system as a smartphone held in landscape mode, their performance improved significantly. This highlights the importance of using a familiar layout for the users. However, expert users may wish to customize the layout to put the most commonly used keys in the most accurate and comfortable regions.

In a text entry system, a user's typing speed is related to the number of errors they correct, as correction takes time away from inputting the desired text. In our study, the corrected error rate was around 15% in the last session. This is somewhat high compared to other text entry systems. These errors are a combination of a user's inaccuracy in touching the correct part of their finger and DigiTouch's inability to accurately sense the intended touch location. For user's touch accuracy, prior work suggests that users are able to accurately perform thumb-to-finger touches [61], but they did not examine whether this accuracy holds up when typing rapidly. Similarly, for DigiTouch's sensing accuracy, a high-accuracy motion capture system, such as OptiTrack[4], can help in decomposing the sources of error.

Though our evaluation of DigiTouch used a fixed keyboard layout, the continuous sensing could enable an adaptive keyboard model. Users with different sized hands or a different range of motion of the thumb may prefer a condensed or expanded layout, for example. With discrete buttons, it is impossible to relocate them or create dynamic touch regions. During the study, several users expressed frustration at the difficulty of pressing a particular key (usually on a finger with four or five buttons). With an adaptive model, these keys could be virtually expanded to make them easier to hit.

Most modern keyboards offer intelligent auto-completion and auto-correction features. Though we evaluated DigiTouch using text entry with auto-complete functionality disabled to enable comparisons with related work, we anticipate that this can significantly improve typing speed and accuracy. To explore this, we built a novel auto-suggestion system that utilizes the pressure sensing capability of DigiTouch. We assign one potential word completion to each finger. A user can trigger a word completion with a force press on the corresponding finger. The high accuracy on the two-level pressure study supports the feasibility of this approach. Though we do not formally evaluate the performance of these features, one of the authors was able to achieve an average speed of 30 wpm using it.

Finally, while we evaluated the QWERTY layout because it was easy to learn and provided a rigorous test of DigiTouch, we note that power users can achieve performance gains by using more advanced keyboard layouts. More commonly used letters can be moved to the outsides of the fingers and can be spaced further

[4]https://www.optitrack.com

apart to significantly boost performance. Less frequently used keys could be relegated to two-touch keys that require chording.

7.6.3 Limitations and Future Work

For wearable systems, the design and construction has a significant impact on user performance. To maximize comfort and minimize bulk, DigiTouch uses a thin one-size-fits-all elastic glove. However, the touch strips we add to the fingers are not elastic. Though all participants found the glove to be comfortable, some participants mentioned that they would have preferred a glove tailored to their own hand size. Anecdotally, we did not see any deterioration in performance for these participants, but in the future, it may be beneficial to design gloves in several sizes to accommodate users with different hand sizes.

Though DigiTouch is designed for wearable AR and VR scenarios, our evaluation is conducted at a desktop computer. Initial pilot testing with a VR system introduced additional factors unrelated to the performance of DigiTouch, such as general unfamiliarity with VR systems and discomfort due to extended use of a head-mounted display. DigiTouch is capable of handling varying resistance of the fabric strips, but it assumes a uniform resistance along the length of the strip at any point of time. This can become problematic near each end, where the wire connection is made on the sides of the finger (Figure 7.4). Because of this connection, the end points are more conductive, which results in a nonlinear region at the extremes. Future versions should be able to account for this with a one-time calibration. DigiTouch enables continuous pressure input. This is slowly becoming popular on consumer smart devices, and perhaps will be even more valuable in the 3D environments of AR/VR systems. In the future, it will be interesting to evaluate the benefits and performance of pressure input, beyond discussing its use for auto-correction in text entry.

To explore the use of DigiTouch in an AR device, we created a wireless version of the gloves that streams data to a Unity application running on a HoloLens. We designed a custom Unity input module that allows DigiTouch to be used with all the standard Unity UI components, including buttons and sliders. Using this module, we constructed two example experience that can be controlled with DigiTouch: a music player and a text entry system. While these experiences highlight the potential of this style of input with AR systems, we leave a formal evaluation to future work.

DigiTouch's continuous position and pressure input may support richer interactions when combined with other sensing. For example, in a sculpting application, a user might configure a brush with color, brush size, and other settings, then force tap it onto a finger so it "sticks." Future taps on that finger would trigger that tool, enabling user-customized interfaces. Using gaze selection, a user might look at a virtual object and drag along the finger to move it closer or farther away; added hand orientation sensing might set the direction of movement. Adding other sensors, such as an inertial measurement unit or flex sensors, could enable even richer interaction that takes advantage of the movement and posture of the hand. Adding vibrotactile haptics could improve feedback and add affordances. For example, it may vibrate when a user slides across a button that can to be pressed.

7.7 Conclusion

DigiTouch is a reconfigurable glove-based input device for wearable computing, particularly head-mounted AR and VR systems. It enables subtle thumb-to-finger interactions by sensing the continuous touch position and pressure of the thumb along the fingers. To achieve this, we present a novel technique using only partially conductive fabric and a two-wire interface on each finger, with a conductive fabric patch on the thumb. Continuous touch tracking enables a set of easily reconfigurable widgets, which can be customized based on user preferences and application needs. To evaluate the performance of DigiTouch in a real-world application, we conducted a longitudinal text entry study using split-QWERTY keyboard. Participants achieved a mean typing speed of 16.5 wpm with high accuracy, showing the feasibility of using DigiTouch for text entry. The subtle, yet always-available input offered by DigiTouch has the potential to enable broader use of AR and VR systems.

8 WEARABLE FINGER TRACKING

8.1 Introduction

While DigiTouch explored interaction techniques for mobile computing, this chapter takes a step back and focuses on the specific hardware and form-factors that may be appropriate for extended use. Restricting ourselves to form-factors that have already been validated, what kind of sensing and tracking can enable continuously available, robust, and expressive interaction?

This chapter describes AuraBand, a precise, millimeter-level finger tracking system. AuraBand consists of two components: a ring and a wristband. The low-power battery-operated ring generates an oscillating magnetic field around the hand. As the user moves their finger and wrist, the relative position and orientation of this field changes. Sensors embedded in the wristband measure these fields at different locations and estimate the pose of the ring with respect to the wristband. By using oscillating magnetic fields, AuraBand is not affected by the Earth's geomagnetic field. AuraBand is effectively a 5 degree-of-freedom (DoF) tracking system—that is, it tracks three positional components and two rotational components (yaw and pitch) of pose. This allows AuraBand to capture flexion/extension and abduction/adduction of both the wrist and finger metacarpal joint. Although the ring can be worn on any finger, it is intended to be worn on the index finger for pointing tasks.

Compared to prior work, our approach leverages common device form-factors—a wristband and a ring—and does not require affixing magnets [52, 20, 61, 19], iron-core coils [21], or magnetometers [20, 61, 19] to the fingertips. AuraBand leverages the insight that an electromagnetic coil is better placed around the finger than on it. Our approach also delivers significant improvements in power, range, portability, and tracking precision.

Electromagnetic tracking has a long history of use in meter-scale tracking applications [80, 149, 150, 116, 108, 125] in a variety of domains. Though precise, electromagnetic tracking has a reputation for being power-hungry and subject to environmental interference. AuraBand overcomes these challenges by explicitly focusing on short-range ($10\,\mathrm{cm}$ to $15\,\mathrm{cm}$) tracking. This range reduces the scope of possible interfering objects and allows the AuraBand transmitter to use approximately 2000x less power than a typical commercial EM tracking system [125].

Sensor-based efforts to track finger motion in space often focus on discrete, gesture-based interactions [48, 176, 134, 195, 7]. Although gesture recognition is useful, wearable input devices must be robust and support different users and contexts of use. To enable this level of precision and robustness, AuraBand supports high-fidelity continuous tracking, upon which gesture recognition or other interactive systems can be built. Because AuraBand leverages physics models, it works out of the box after a one-time factory calibration with ground-truth tracking data from a motion-capture system. We demonstrate that at runtime, AuraBand can track finger motion across different users and despite changes in how the device is positioned on the wrist and finger. In cases where additional accuracy is desired, the calibration can be improved with just

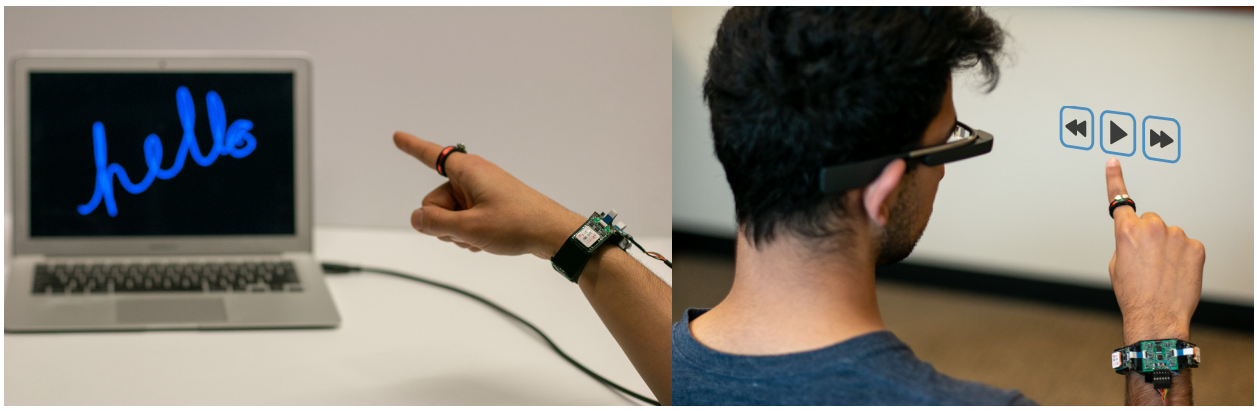

Figure 8.1: AuraBand is 5-DoF electromagnetic tracker that enables precise, accurate, and fine-grained finger tracking for AR, VR and wearable applications. Left: A user writes the word "hello" in the air. Right: Using AuraBand to control a music application on smart glasses.

60 sec of additional data.

AuraBand offers a flexible solution for tracking the finger in either direct or indirect pointing tasks. We envision that AuraBand can be used either as a standalone input device, or in tandem with a wrist-tracking solution. For standalone scenarios, AuraBand offers a rich input source for smartwatches or smart glasses. With AuraBand, a user can provide input using their finger and wrist while keeping their arm motionless at their side or resting on a table. This capability is particularly useful in public settings, when large hand motions would be distracting and socially unacceptable.

AuraBand can also be used to track the absolute position of the finger in head-space with respect to the user's vision. In mixed-reality scenarios, precise hand and finger tracking enables many compelling applications. For example, buttons, sliders, and interactive widgets can be placed in the air, on environmental surfaces, or on the body. AuraBand can be operated in concert with a head-mounted camera that tracks the wrist with respect to the head using fiducials on the wristband.

Our primary contributions include:

1. A hardware architecture for a ring and wristband that enables low-power finger tracking in a compact form-factor.

2. Two tracking algorithms, including a physics-based iterative approach and a closed-form neural network approach to 5-DoF pose estimation.

3. A system characterization and user evaluation demonstrating mean tracking a resolution of 100 µm and dynamic accuracy of 4.4 mm on a session-independent task.

8.2 Theory of Operation

8.2.1 AuraBand Physics

Magnetic tracking is generally realized either with magnetometers to track a permanent magnet [7, 20, 61] or with inductive coils to track an alternating current (AC) electromagnet [149, 150, 181, 185, 125, 108, 131, 80]. Static magnetic fields created by a permanent magnet or inseparable from the Earth's geomagnetic field—an effect which becomes critical when operating at distances beyond a few centimeters. To isolate the signal of interest, AuraBand uses AC electromagnetic coils to generate a magnetic field at a particular frequency (32 kHz). Maxwell's equations state that an AC magnetic field is generated when an AC electric current is passed through a wire coil. As depicted in Figure 8.2, AuraBand uses a wire coil wrapped on the ring to produce an AC magnetic field around the hand. According to Faraday's law, a voltage is induced in the other sensor coils in the presence of an AC magnetic field. The induced voltage is proportional to rate of change of magnetic flux through the coil. For a field generated from the ring, the magnetic flux and resulting voltage signal will change as a function of the position and orientation of the sensor coils with respect to the transmitter coil. More specifically, if the sensor coil is aligned with the field (i.e., the normal vector to the coil is aligned with the field), then the magnitude of the induced voltage will be maximized. As the coil rotates away from the field, the induced voltage decreases to zero. If the coil continues to rotate even further, the voltage acquires a 180° phase shift that manifests as a negative amplitude.

Traditional 6-DoF electromagnetic trackers use a 3-axis transmitter and a 3-axis sensor coil, which is difficult to achieve in a ring form-factor where size and battery life are paramount. However, a ring is a convenient form-factor for a single-axis air core coil that wraps around the finger. Using a single-axis coil makes the entire system insensitive to changes in the roll of the transmitter along the magnetic axes; conveniently, such movement is not physically relevant in finger-tracking scenarios. To compensate for the lack of three transmitter axes, AuraBand leverages three 3-axis sensor coils embedded at known locations within a wristband. By measuring the magnetic fields at different points in space, AuraBand reconstructs the 3-DoF position and 2-DoF orientation of the ring. Figure 8.2 illustrates the configuration of the AuraBand transmitter and sensors.

8.2.2 Magnetic Field Model

We construct a physics-based simulation to model the behavior of our system under different configurations, such as changing the number or placement of sensor coils. Because the ring diameter is much smaller compared to the distance between the ring and wristband, we use standard magnetic field equations to model the ring as a dipole emitter. The model seeks to estimate the magnetic field ($\vec{B}_s$) measured at each sensor coil (s) in the wristband. Because the transmitter is embedded within the object to be tracked, we first conduct a series of coordinate system transformations. In this chapter, we adopt the notation $\vec{B}_s$ to represent the $\vec{B}$ vector in the s coordinate frame. We represent a rotation using $\mathbf{s}_w$ to represent a transformation from the w frame to the s frame. In practice, these are implemented using quaternions, but other representations would be appropriate as well.

Let d be the dipole coordinate frame with the magnetic axis oriented in the z-direction. This is coincident with the ring coordinate frame. Let w be the wrist coordinate frame, which contains multiple sensors s_i

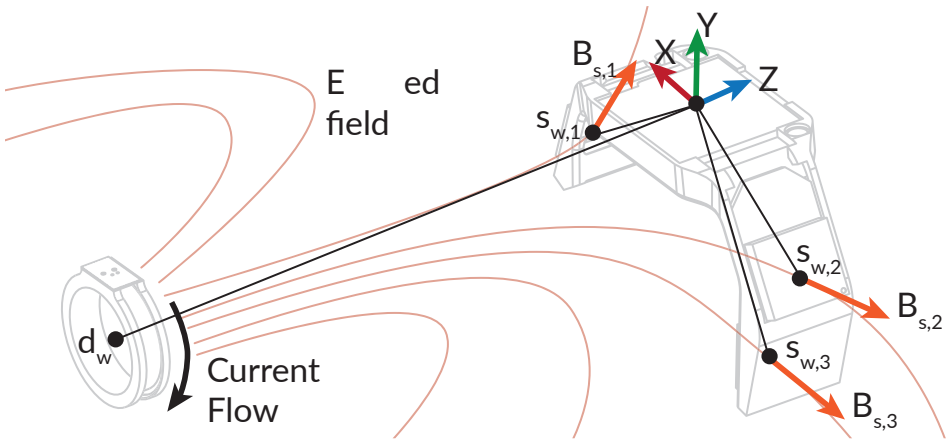

Figure 8.2: AuraBand uses a wire coil wrapped on a ring to produce an AC magnetic field around the hand which is measured by three 3-axis coils embedded in a wristband.

at positions $\vec{s}_{w,i}$ and orientations $\mathbf{s}_{w,i}$, as depicted in Figure 8.2. Given a position of the ring ($\vec{d}_w$) and the geometry of the sensors ($\vec{s}_w$), we first compute the position of the sensor in the dipole frame:

$$\vec{s}_d = \mathbf{d}_w \times \left(\vec{s}_w - \vec{d}_w\right) \tag{8.1}$$

We then use standard magnetic dipole equations derived from the Biot-Savart law under quasistatic assumptions to compute the magnetic field at each sensor location ($\vec{B}_{d,i}$):

$$B_x = \frac{3xz}{r^5} \qquad B_y = \frac{3yz}{r^5} \qquad B_z = \frac{3z^2 - r^2}{r^5} \tag{8.2}$$

Finally, we rotate once more to compute the magnetic field in the sensor frame of reference:

$$\vec{B}_s = \mathbf{d}_w^* \times \vec{B}_d \tag{8.3}$$

8.2.3 *Number of Sensors Required*

To inform the design of AuraBand, we conducted a preliminary analysis using this model in which we varied the number of sensor coils. Choosing a sensor coil configurations requires balancing performance versus size and power. Tracking the ring is at least a 4-DoF task, since both the wrist and finger metacarpal joints have two degrees of freedom. As a result, a robust approach requires at minimum two coils; each coil measures a three-dimensional field vector from a single emitter, so two coils would provide six sensor values. In practice, due to the potential for small shifts of the band on the wrist, AuraBand tracks as a 5-DoF task (x, y, z, pitch, yaw), which we hypothesized might be difficult with just two sensor coils. A configuration is considered robust if, for a set of observed sensor values, there is a unique position and orientation within reasonable bounds that explains those observations.

To explore the design space of coils, we first choose a test point, representative of a typical ring position and orientation, and use the physics model described previously to compute the magnetic field as observed from each sensor. We then compare this observation to a dense sampling of positions and orientations. The x, y, and z axes are defined according to the wrist frame in Figure 8.2, with the finger generally pointing along the $-Z$ axis. For orientation, we define θ as pitch (in the extension/flexion direction) and ϕ as yaw (in the adduction/abduction direction) For position, we sample at $2\,\mathrm{mm}$ intervals within the bounds of $-40\,\mathrm{mm} < x < 40\,\mathrm{mm}$, $-130\,\mathrm{mm} < y < 80\,\mathrm{mm}$, and $-140\,\mathrm{mm} < z < -50\,\mathrm{mm}$. For orientation we use pitch, θ and yaw, ϕ, and Euler angles to generate a quaternion orientation at 2° intervals within the bounds $-90^\circ < \theta < 90^\circ$ and $-60^\circ < \phi < 60^\circ$. These bounds correspond to typical bounds of the ring for typical bone lengths and range of motion.

Figure 8.3 illustrates an example result of this analysis for a single target point. The points in each plot indicate locations that can produce a set of sensor measurements similar to that produced at the target point under any tested rotation. This means that for a given point in the plot, there is some ring orientation at that position that could be confused for the target point, shown in red. The color indicates the measure of similarity to the target point.

As expected for a device with only a single sensor, there is a large confusion volume along all dimensions. Interestingly, there is a defined minimum for a 2-sensor setup, but the problem is ill-conditioned along some directions. This suggests that a 2-sensor device might suffice with the proper constraints, but any noise could result in significant errors. The 3-sensor setup produces a well-conditioned minimum, justifying our use of three coils around the wristband. Although this particular example does not reveal much about optimal sensor placements, robustness is maximized when the sensor coils placements are maximally spread apart.

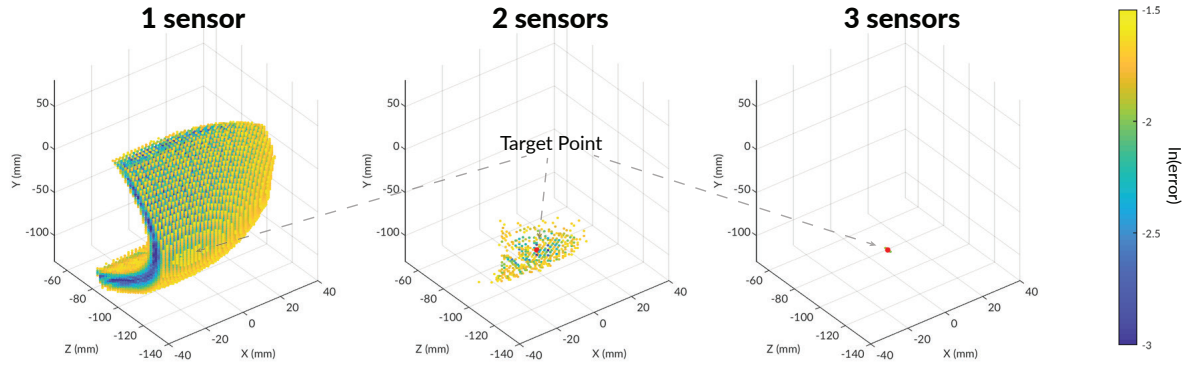

Figure 8.3: By embedding three sensor coils in the wristband, AuraBand achieves a well-conditioned minimum that makes AuraBand robust. As depicted, having one or two coils will result in confusing other points with the target point.

8.3 AuraBand Hardware

The AuraBand system consists of a ring-worn device that generates an AC magnetic field and a wristband device with three embedded sensors that measures the resulting fields. Figure 8.4 depicts the AuraBand system components. The following sections describes the AuraBand hardware, explores the hardware's capabilities, and enumerates design challenges.

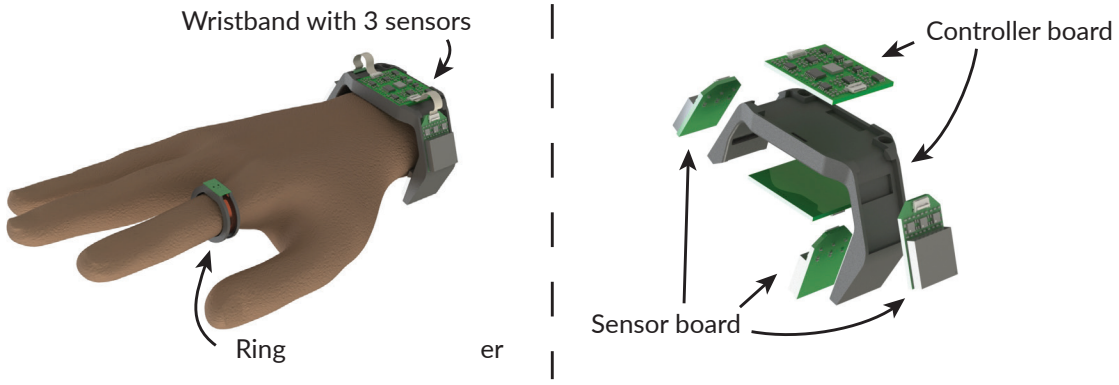

Figure 8.4: AuraBand contains two controller and three sensor boards embedded in a wristband and a ring worn device.

8.3.1 Ring

In designing AuraBand, we pursued a design that would resemble device form-factors that users are already accustomed to in order to present a plausible path forward for everyday use. For the ring-based transmitter, rather than affixing an iron-core coil [21] or permanent magnet [19, 20] to the top of the finger, we designed a custom electromagnet transmitter compatible with a ring form-factor. In doing so, we were forced to put minimal digital electronics on the ring to ensure a slim form-factor and minimize power consumption. AuraBand's ring consists of a single-axis low-profile transmitter coil that emits a magnetic field oscillating at 32.768 kHz. The generator coils consists of 800 turns of 42 AWG magnet wire wound around a 3D-printed ABS ring with a diameter of 20 mm. The inductance of the coils is approximately 15 mH. The number of turns and wire gauge was chosen to maximize the magnetic field strength while minimizing DC resistance, skin effects, and weight.

For the transmitter electronics, a small custom PCB ($6.9\ \mathrm{mm} \times 11.4\ \mathrm{mm}$), sits horizontally on top of the 3D printed ring. Figure 8.5 (right) shows a block diagram of AuraBand's transmitter. The transmitter uses a surface-mounted oscillator (ACZ-32.768) to generate a $32.768\,\mathrm{kHz}$ square wave. This frequency was chosen because it is a commonly used frequency for low-frequency clocks on microcontrollers and there is an abundance of integrated chips that produce this frequency. We use a capacitor network to tune the coil's impedance at $32.768\,\mathrm{kHz}$ in order to maximize the transmit power of AuraBand. In short, the series capacitor stores charge that can be drawn through the inductive coil on each cycle of the waveform. This

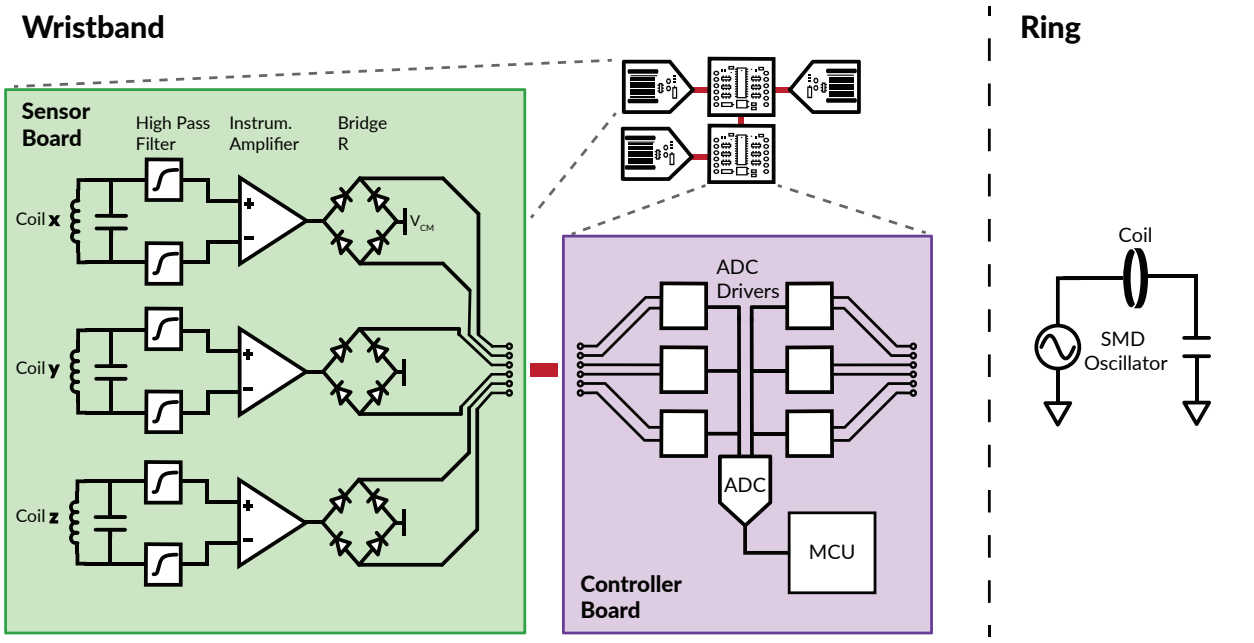

Figure 8.5: Block diagram of AuraBand. The ring generates a magnetic field at a particular frequency which is measured by the sensor boards. They send these signals to the controller board where where an ADC samples the data and communicates with a host computer using an ultra low power MCU.

effectively reduces the operating voltage required to generate a given field strength. For a detailed discussion of the effect of these resonant networks on tracking systems, see [65]. The transmitter is powered with two $12\,\mathrm{mA\,h}$, $1.55\,\mathrm{V}$ coin cell batteries with a diameter of $4.8\,\mathrm{mm}$ and a height of $2.1\,\mathrm{mm}$.

8.3.2 *Wristband*

The AuraBand wristband consist of three sensors and two controller boards as depicted in Figure 8.5 (left). At a high level, the sensor boards each measure a magnetic field while the controller boards convert these measurements to digital signals and communicate them to a host computer.

The sensor boards measure the magnetic field generated by the ring using an off-the-shelf three-axis orthogonal receiver coil (Grupo Premo 3DC15). The signal from each axis is fed to an amplifier (INA826) with a gain of $44\,\mathrm{dB}$. We then use a low-noise and low-voltage drop Schottky diode network (SMS7630) in a full-wave bridge rectifier configuration to demodulate each of the channels. We have used this passive configuration to demodulate the field in order to save power. It is worth mentioning that using this method, we only have access to the fields' magnitudes; AuraBand does not know whether each channel is in- or out-of-phase with the transmitter. Because of the hand's kinematics, most of the channels' signs remain constant, so there is not much information lost. In Section 8.5, we show that AuraBand can still track without knowing the exact signs of the fields.

These magnitude signals are passed to a controller board with a 10-wire FFC cable. Each controller board

supports two sensor boards. By vertically stacking two controller boards within the center of the wristband, AuraBand supports up to four sensors. Based on the analysis in Section 8.2.3, three coils are sufficient for tracking, so we connect two of the sensors boards to the top controller board and one to the bottom board.

On each of the controller boards, there is an SAR analog-to-digital converter (AD7265) that samples six differential signals simultaneously at $31.25\,\mathrm{kHz}$. Each channel on the controller board has an ADC driver (LTC6363) to achieve high precision, resolution, and throughput. Finally, the sampled data is passed to an ultra-low power MCU (MSP430FR2422) for further processing and communication. Every eight ADC readings, the bottom controller board sends the sampled data to the top controller board where the data is collected and sent to a host computer over USB at $472\,\mathrm{kHz}$. Figure 8.5 summarizes the analog signal processing of AuraBand.

8.4 Calibration

AuraBand's tracking algorithms have, at their core, a generative physics model, described in Section 8.2.2. This can be used to generate synthetic data for model training or used directly for model-based tracking. However, before the physics model can accurately describe the AuraBand measurements, we must learn a number of parameters, such as channel gains and precise sensor positions. In a precise manufacturing environment, these parameters can likely be set based on an analysis of the system geometry and electronics. However, in our prototype, due to the tolerances of 3D printed parts and our selection of electronic components, we opt to learn these parameters empirically by collecting and using data from an optical motion capture system. In the following sections, we describe our data collection setup, the calibration model, and learning procedure. Figure 8.6 shows a high-level overview of the calibration process and how it relates to the runtime pose estimation task.

8.4.1 Data Collection Setup

We use a 10-camera[1] optical motion capture system (calibrated accuracy of $0.1\,\mathrm{mm}$) to track the ground truth position and orientation of the wrist and ring. To facilitate this, we place IR retroreflective markers on both devices, as shown in Figure 8.7. The motion capture system reports the pose of the wristband $(\vec{w}_m, \mathbf{w}_m)$ and ring $(\vec{r}_m, \mathbf{r}_m)$ in room-space at $240\,\mathrm{Hz}$. Software in Python on a PC logs both the motion capture data and AuraBand sensor data to disk for offline analysis.

We first preprocess all of the motion capture data to align the rigid body coordinate space to our coordinate system defined by the device geometry, as shown in Figure 8.2. Next, to reconstruct the relative pose of the ring in wrist space, we apply another coordinate system transformation.

$$\vec{r}_w = \mathbf{w}_m \times (\vec{r}_m - \vec{w}_m) \tag{8.4}$$

$$\mathbf{r}_w = \mathbf{r}_m \times \mathbf{w}_m^* \tag{8.5}$$

Finally, we synchronize the sensor stream from AuraBand ($472\,\mathrm{Hz}$) with the ground truth ring pose stream ($240\,\mathrm{Hz}$). We align the two streams by using the ring pose to roughly estimate the magnetic fields, as

[1] OptiTrack Prime 13

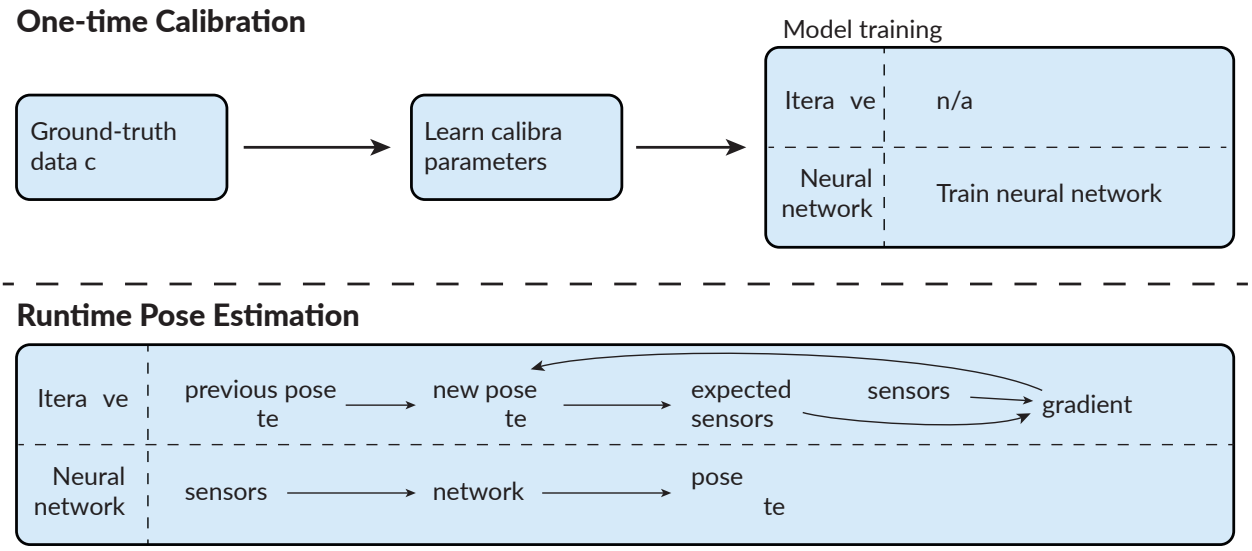

Figure 8.6: The AuraBand system needs a one-time factory calibration to match the synthetic data to its measurements. In real-time, the iterative model uses this calibration to refine the pose estimation and the neural network model regresses directly to pose.

described in Section 8.2.2, and comparing these fields to the AuraBand sensor values. We then resample the AuraBand signal through interpolation to 240 Hz, align with the ground truth signal. During this process, we drop any frames in which the motion capture system lost track of either the ring or wristband.

8.4.2 Sensor Model Parameters

Each sensor pipeline in AuraBand transduces the incident magnetic field to a digital value. For a given ring pose, we need to be able to estimate the values reported by each sensor. Doing so accurately requires additional calibration parameters that define the *sensor model*, which describes how a given pose manifests as sensor measurements. These parameters model effects like the sensitivity of each coil and the relative positioning of the sensor coil on the PCB. Altogether, they capture offsets between the motion capture positions and the effective magnetic origins as well as effects of the AuraBand analog signal processing pipeline. Below, we briefly outline these parameters and how they fit into the AuraBand sensor model.

Coordinate System Offsets

Due to manufacturing tolerances, the exact position of each sensor with respect to origin reported by motion capture may not exactly align with the computed values from the CAD design. Consequently, we define offset positions and orientations to refine each sensor pose in the wrist frame. These slack terms

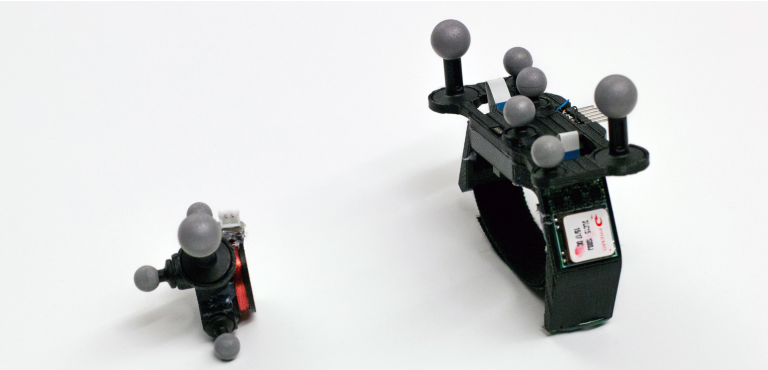

Figure 8.7: We place retromarkers on the ring and wristband to get the ground truth measurement from an optical capture system.

are expected to be small — on the order of a couple of millimeters or degrees.

$$\vec{s}_w = \vec{s}_{w,CAD} + \vec{s}_{w,offset} \tag{8.6}$$

$$\mathbf{s}_w = \mathbf{s}_{w,offset} \times \mathbf{s}_{w,CAD} \tag{8.7}$$

We use a similar technique to define the dipole coordinate frame with respect to the ring coordinate frame.

$$\vec{d}_w = \vec{r}_w + \mathbf{r}_w \vec{d}_r \tag{8.8}$$

$$\mathbf{d}_w = \mathbf{d}_r \times \mathbf{r}_w \tag{8.9}$$

Modeling Analog Signal Chain

We then use the model in Section 8.2.2 to compute B_s, the magnetic field in the sensor reference frame. However, the sensors and amplitude demodulation pipeline do not represent a perfect measurement of the magnetic field. For one, the rectifier-based demodulation scheme only provides an unsigned estimate of the field strength along each axis. Moreover, each channel has a slightly different gain, due to different sensitivities and manufacturing tolerances of both the sensor coil and amplifiers. We also model the effects of noise and the diode forward voltage drop. Specifically, we estimate the sensor measurement (x_i) as a function of the field ($B_{s,i}$) and the channel gain (g_i), noise (n_i), and bias (b_i).

$$x_i = \sqrt{(g_i \times B_{s,i})^2 + n_i^2} - b_i \tag{8.10}$$

Altogether, this model includes 23 parameters that must be learned for each sensor coil.

8.4.3 *Learning Sensor Model Parameters*

To learn these parameters, we collected a dataset that captures a wide range of motion of the ring. One of the authors wore the wristband while using the other hand to move and rotate the transmitter freely within

in a volume around the hand. During this process, the ring and wrist pose are captured by the motion capture system and aligned as previously described. The data was explicitly collected without wearing the ring so that this dataset contains 6-DoF motion of the ring. Otherwise, the data would constrained by the 4-DoF wrist and finger kinematics of a particular user and it may be possible to overfit these parameters to that user.

We took a random sample of 30k frames from this dataset to use for learning and formulated the problem as a non-linear optimization problem, which we solve using Ceres Solver [3]. We learn the 23 parameters for each sensor coil independently. In doing so, the solver seeks to find the parameter set which minimizes the difference between the observed sensor values and the estimated sensor values, computed according to the observed pose and the sensor model. The solver uses the Levenberg-Marquardt trust region algorithm for minimization.

Upon convergence, this results in a set of parameters that can be used to estimate the sensor measurements as a function of pose. Figure 8.8 shows the correlation between the sensor estimates and actual measurements on this dataset before and after applying the sensor model presented here. Before applying the sensor model, the estimates have a 0.564 Spearman's rank-order correlation with the actual measurements. After applying the sensor model, the correlation increases to 0.995, validating the effectiveness of this model.

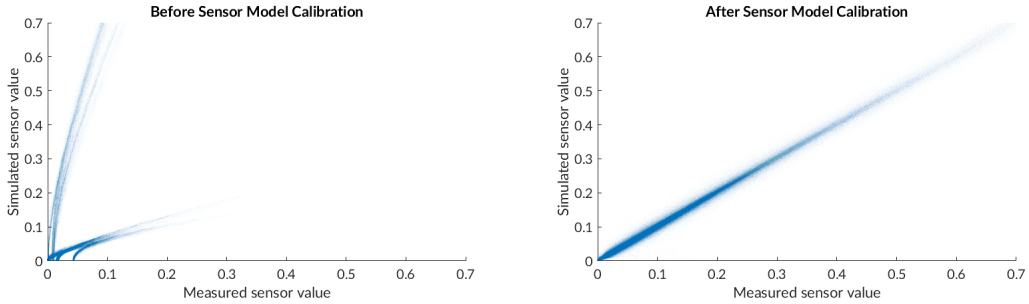

Figure 8.8: The one-time factory calibration process is necessary to match the synthetic data to the data measured by AuraBand.

8.5 Tracking Algorithm

Taken together, the magnetic field model and sensor model represent a forward model of the system—they estimate the sensor values given the ring pose. To track the ring, we must solve the reverse problem—estimating the ring pose given the sensor values. We implemented two different tracking algorithms to solve this reverse problem and estimate the 5-DoF pose of the ring. The first is an iterative optimization-based solution and the second is a neural network that approximates a closed-form solution to the reverse problem.

8.5.1 Approach 1: Iterative Model

The iterative model is similar to the method of learning calibration parameters discussed in Section 8.4.3, but operates on each frame separately. We use a non-linear optimizer [3] with the Levenberg-Marquardt algorithm to iteratively find the most likely pose. When a new set of sensor data arrives, the solver minimizes the error between the observed sensor values and the sensor values predicted by the forward model (Equation 8.10). It is allowed to run for a maximum of 100 iterations. For the first frame, the initial pose state is set to a default pose with the index finger pointing forward without bending. For subsequent frames, the solver uses the solution to the previous frame as the initial state. Further improvements in speed and accuracy may be possible by using a Kalman filter to proactively estimate the next ring pose.

Because the sensor measurements in the presence of a weak field fundamentally have a lower signal-to-noise ratio (SNR), we do not want these noisy measurements to significantly effect our estimates. Moreover, because Equation 8.10 has a minima when the field is null, there is the potential to get stuck in this minima. To reduce the impact of these effects, for any of the nine channels in which the measured magnetic field is extremely weak, we simply drop this term from the cost function.

To understand how the specific sensor calibration parameters affect tracking performance, we created a variant of the iterative tracking model that refines the calibration parameters based on two minutes of data from that session. This accounts for drift in parameters like the noise in the analog signal change, which can be impacted by environmental factors like temperature. We refer to the original session-independent iterative as "precalibrated" and the session-dependent model as "recalibrated". Differences between the "precalibrated" and "recalibrated" models are likely due to manufacturing tolerances of this particular prototype. With additional engineering effort, we anticipate any performance gap here to be improved.

8.5.2 Approach 2: Neural Network-based Tracking

As an alternative to the iterative model, we used closed-form approximations to the reverse problem that are optimized for use on low-power devices. Specifically, we trained a neural network to predict both the 3 DoF position and 2 DoF orientation of the AuraBand ring directly from the magnetic measurements.

Generating Training Data

To eliminate any dependency on runtime training data, we train the network entirely from data generated by our simulated magnetic and sensor models. This gives us precise control over the sampling distribution of the training data and eliminates issues of over-fitting to a particular user's hand. We randomly generated different ring poses within a volume relative to the wrist that spans possible positions of the ring for most adults. Specifically, assuming the coordinate system defined in Figure 8.2, the generated data covers a volume defined by $-70\,\mathrm{mm} < X < 90\,\mathrm{mm}$, $-125\,\mathrm{mm} < Y < 60\,\mathrm{mm}$ and $-150\,\mathrm{mm} < Z < -60\,\mathrm{mm}$. We also add a random 3-DoF rotation to each point to complete the dataset of random poses.

We then use the sensor model to estimate the values of AuraBand's sensors at each pose. To improve performance, we further refine the dataset of poses to eliminate infeasible poses during normal operation by considering the direction of each field. Since AuraBand uses 3 three-axis receivers and each axis could be in or out of phase with the transmitter, there are $2^9 = 512$ possible combination of phase states for a

given frame. However, due to the spatial geometry of our sensors and the kinematic structure of a hand, the majority of these states are not feasible during normal use. For example, because the ring always lies in front of (negative z-direction) the wristband, all of the z-channels of the sensors will have the same phase. In fact, after reviewing the generated data we found that only ten of these 512 combinations are feasible. We drop any data from the generated dataset that does not match one of these ten feasible phase states. Since training is done using simulated data, one could easily change these constraints or even add another state and train a larger network if the problem demands it. After this culling process, the dataset consists of 166,465 points.

NN Training

We propose to use two computationally simple models to regress to a position vector and the 2-DoF orientation of the ring. Both networks have a single hidden layer of 128 nodes to fit a function that maps the nine observed sensor values to a 3-dimensional position vector and a 3-dimensional direction vector of unit length. We use a direction vector to specify orientation in order to remain invariant to the roll of the ring. Training using the synthetic dataset is performed in MATLAB with the Levenberg-Marquardt algorithm. With CPU training on a single core of a Xeon E3-1240 processor, training takes about six hours. Mean training error was $3.06\,\mathrm{mm}$ for the position model and $5.45°$ for the orientation model. Orientation error was calculated by computing the angle between the two direction vectors.

8.6 System Evaluation

8.6.1 Tracking Accuracy

User Evaluation Procedure

We evaluate the tracking accuracy of each of these models using data collected from users who we invited to try AuraBand. We do this to provide a realistic estimate of real-world performance under challenging conditions like different hand size, shape, flexibility, slippage of the devices on the hand and finger, and changes in environmental factors. We recruited 14 participants (5 M, 9 F) with different hand sizes to wear AuraBand while exercising the full range of motion of their wrist and fingers. The data collection was carried out in the same 10-camera motion capture lab used to calibrate the sensor model. The researchers helped the participants put on the wristband and placed the ring on the index finger of the right hand. To ensure no issues related to battery life, the ring was powered with a slightly larger $105\,\mathrm{mA\,h}$ LiPo battery that they held within their hand. As before, markers were placed on an attachment to the ring and wristband to facilitate optical tracking. As showed in Figure 8.9, we placed four additional markers on the top side of the hand for debugging and visualization purposes. The participants were asked to freely and naturally move their wrist and finger for 10 minutes, while being sure to exercise all possible joint motions. During this time, the ground-truth pose and the magnetic sensors data were recorded by a Python program.

After processing the data, we dropped data from two participants due to poor quality of the optical tracking from self-occlusions. Altogether the remaining data consisted of more than two hours of data and over 1.7 million synchronized data points.

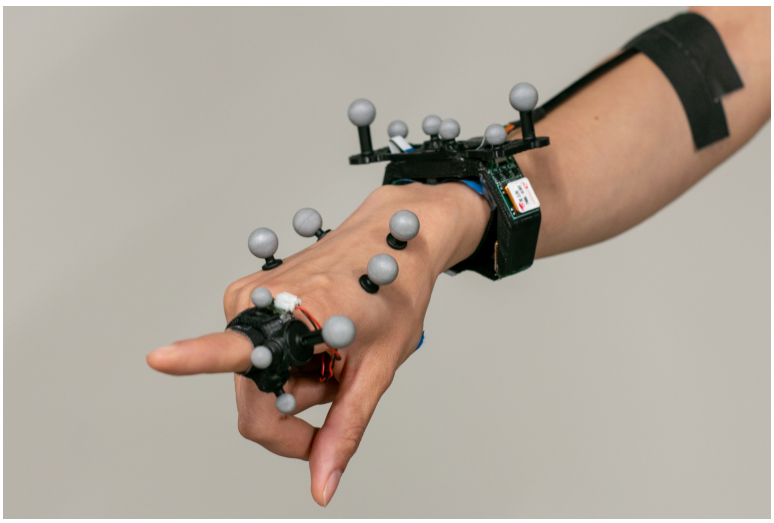

Figure 8.9: Data collection setup. IR retroreflective markers are placed on the ring, palm, and wristband to facilitate tracking. The user moves the index finger and wrist while measurements are collected. AuraBand streams data to the PC over a USB connection.

Table 8.1: Shows the position and orientation error among 12 participants using iterative and neural network models

	NN Model	Precalibrated Iterative Model	Recalibrated Iterative Model
Mean X Error (mm)	2.85	1.65	0.50
Mean Y Error (mm)	4.18	3.41	1.21
Mean Z Error (mm)	1.80	1.14	0.51
Mean Position Error (mm)	6.07	4.41	1.53
Mean Orientation Error (degrees)	8.35	4.65	1.69

Results

We compare the tracking accuracy for both the iterative and neural network models as well as the session-dependent recalibrated iterative model. Table 8.1 shows the mean accuracy for the pose of the finger using each algorithm. The full error distribution is shown in the CDF in Figure 8.10 that aggregates data across all twelve participants. These results show a mean tracking error of $4.41\,\mathrm{mm}$ for the precalibrated iterative model and $6.07\,\mathrm{mm}$ for the neural network model. The error drops to $1.53\,\mathrm{mm}$ after fine-tuning the sensor model for each session.

For orientation, the iterative model tracks the forward direction of the finger with a mean error of $4.65°$ and the neural network tracks with a mean error of $8.35°$. The orientation error drops to $1.69°$ with the recalibrated model. The full distribution of orientation error for each model is shown in Figure 8.11.

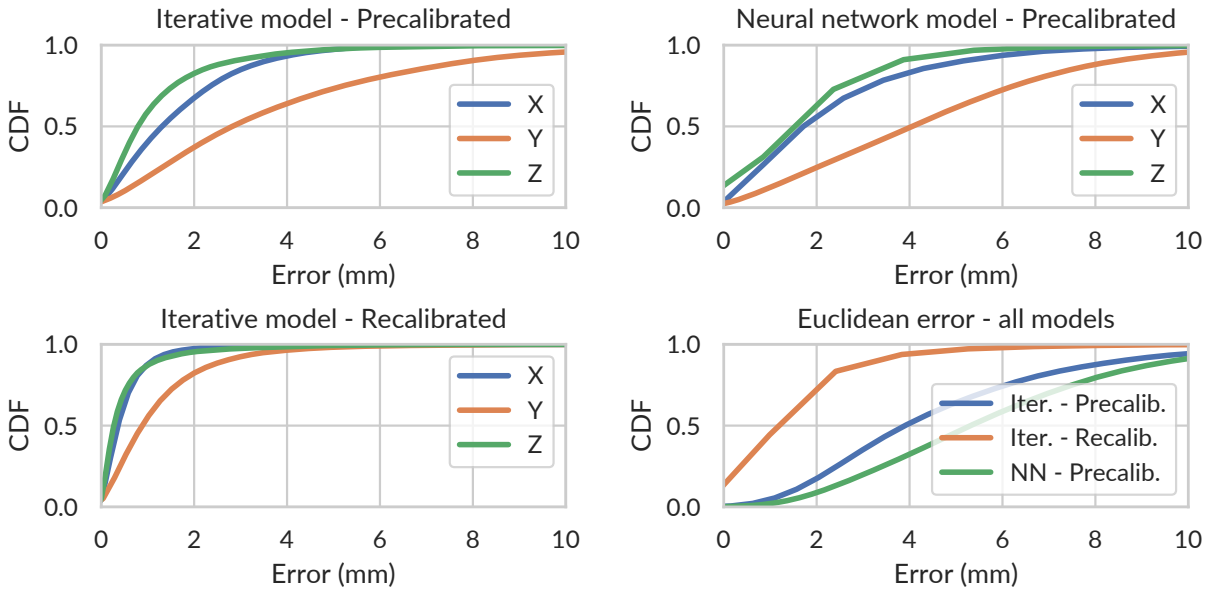

Figure 8.10: CDF of 3D position tracking among all participants using the iterative model vs the neural network model. The best accuracy is achieved by using the recalibrated iterative model

While accuracy is an important measure of tracking performance, the ability to track relative motion is essential if using AuraBand as an input device. Figure 8.12 shows the ground truth and estimated position over a representative few seconds of two representative sample traces. The sample trace on the left, has a below-average mean error on the iterative model of $2.56\,\mathrm{mm}$. The trace on the right has above-average error at $4.99\,\mathrm{mm}$. Note that even in cases where the error is relatively high, the relative motion still tracks the ground truth motion. These traces do not have any additional Kalman filtering applied to them.

To better illustrate the dynamics of each model, Figure 8.13 shows a highly zoomed in trace of the x-direction only for a representative $200\,\mathrm{ms}$ window. The dark lines indicate estimates that have been smoothed with a Kalman filter. The lighter lines represent the raw estimates from each model. This shows that the relative pose tracks well, even over timescales of tens of milliseconds. This pattern holds for the neural network model as well.

8.6.2 *Resolution*

Resolution is an important characterization for a sensor system that quantifies the smallest detectable change in the measure of interest. The use of high-precision electromagnetic sensors represents a significant advantage for AuraBand over other wearable sensing systems and enables its ability to capture small and subtle motions.

We used a motorized linear stage[2] to quantify the resolution of the AuraBand hardware. The stage is

[2]PI VT-80

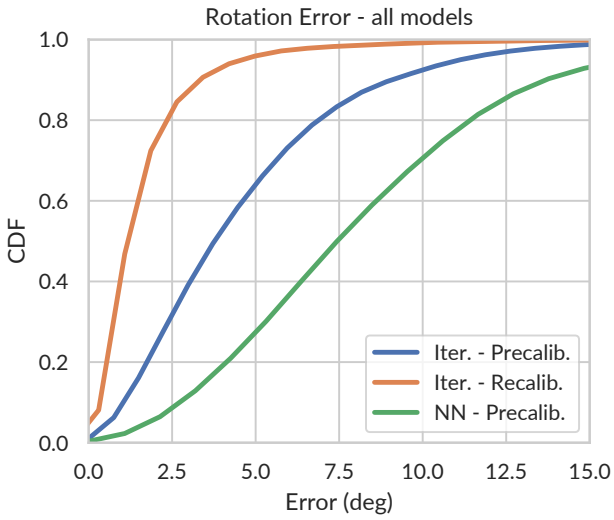

Figure 8.11: CDF of orientation tracking among all participants using all models

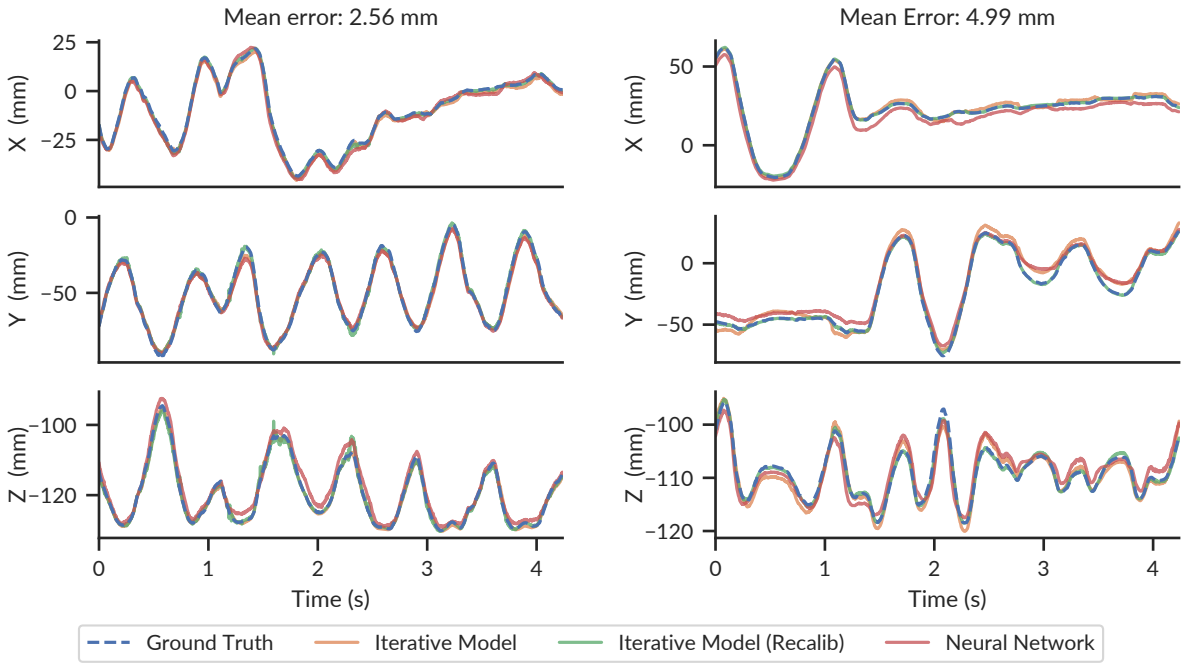

Figure 8.12: A representative few seconds of 3D positional tracking for two of the participants. While the tracking error is relatively higher for the one on the right, the relative motion still tracks the ground truth motion.

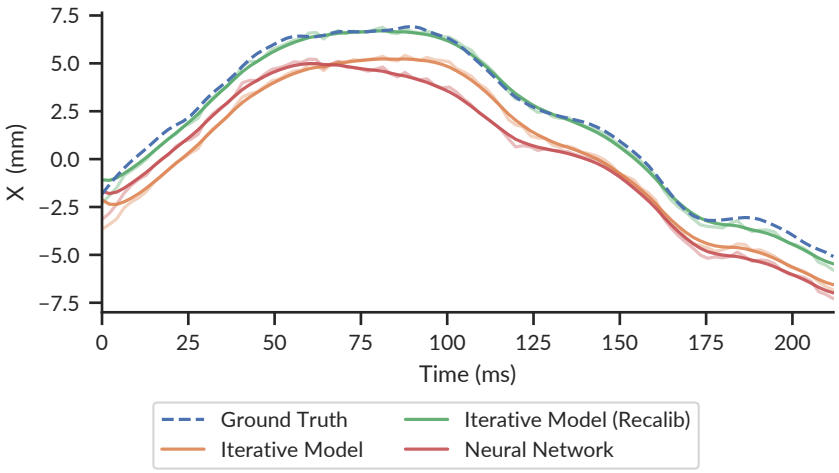

Figure 8.13: A zoomed in trace for the x-direction. AuraBand leverages a Kalman filter to smooth out the estimated position. Even when the neural network has a static offset, the relative motion is preserved.

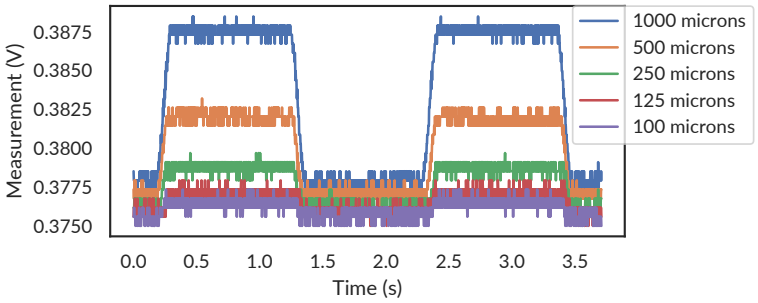

Figure 8.14: The AuraBand hardware is sensitive to 100 microns of movement. A single channel of one of the sensors depicted as the transmitter is moved by a motorized linear stage.

software-controllable and has a repeatability of $10\,\mu m$ and a step size of $0.2\,\mu m$. We affix the ring to the stage and place the wristband $125\,mm$ away, a typical operating distance during normal use. The devices were oriented such that most of the magnetic flux is oriented in the z-direction of the wristband. We programmed the stage to move back and forth (in the z-direction) in variable step sizes ranging from $100\,\mu m$ to $1\,mm$. Figure 8.14 shows the values measured in one of the sensor coils during this procedure. No filtering was applied to this data. From this figure, one can observe that a step size of $1\,mm$ is clearly visible. As the step size decreases to $100\,\mu m$, the different in sensor reading approaches the size of a single bit of the ADC.

We note that because the device is capable of sampling at speeds much faster than a typical user interface would require, the resolution could likely be significantly improved through filtering. We also note that because we do not have a ground truth measure of the AC magnetic field strength, the resolution here depends on the specific position at which it is measured. Nonetheless, this is a representative example that

illustrates the sensitivity of the hardware platform.

8.6.3 Power

We measured the power consumption of both the ring and wristband of AuraBand using a USB oscilloscope to measure the voltage drop across a series resistor of values $9.8\,\Omega$ and $109.5\,\Omega$ for the ring and wristband respectively. The ring consumes only $715\,\mu\mathrm{A}$ of current ($2.34\,\mathrm{mW}$) and the wristband consumes $22.2\,\mathrm{mA}$ ($73.3\,\mathrm{mW}$). Two $12\,\mathrm{mA\,h}$ batteries that fit comfortably on the ring PCB allow the ring to operate continuously for about 17 hours. A $2\,\mathrm{W\,h}$ battery would allow the wristband to continuously operate for more than a full day. In practice, such a system could have a sleep functionality so that the transmitter and processing circuitry do not need to be on when not in use.

8.6.4 Speed and Compute

Although the analysis presented in this work was performed offline, the algorithms used for pose estimation were designed with real-time operation in mind. The iterative approach performs the best, but requires the most compute power. However, despite the iterative nature of the algorithm, it easily exceeds realtime speeds. When fed each data frame sequentially, the algorithm operates at speeds exceeding $1\,\mathrm{kHz}$ on a single core of a Xeon E3-1240 processor. We anticipate that, with some care, it is possible to port this algorithm to a mobile processor.

However, in scenarios where compute is at a premium, such as on a mixed-reality HMD, we designed the neural network approach for maximum efficiency. The network contains only a single layer with 128 hidden nodes. Runtime position and orientation estimation consists of just two matrix multiplications for each (9×128 and 128×3).

8.6.5 Interference

Like all electromagnetic tracking systems, AuraBand's performance will degrade in the close proximity to metallic, particularly ferromagnetic, objects. Unlike outside-in solutions like Polhemus [125], the AuraBand transmitter and receiver are in relative close proximity, which significantly reduces the scope of possible interference sources.

To quantify the effects of large metallic objects on AuraBand, we measured the effect of two commonly used devices, a smartphone and a laptop, that are likely to be in close proximity to AuraBand during operation. The transmitter and receiver are placed on a flat space. One of the researchers brought the devices close to the wristband and ring one at a time. We observed changes in the received signal at a distance of about $15\,\mathrm{cm}$ for both devices.

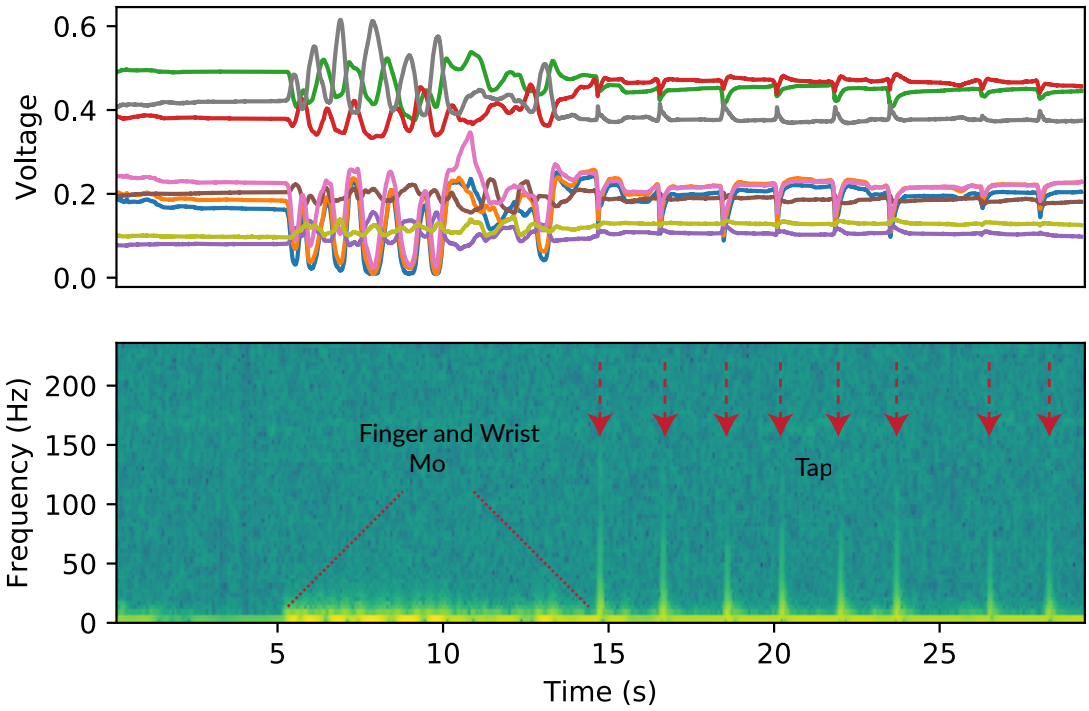

Figure 8.15: AuraBand is capable of measuring high-speed events such as taps. A user performs multiple taps after moving their finger in the air. (Top): nine raw sensor measurements. (Bottom): Spectrogram of the recorded data.

8.7 Additional Functionality

Having the ability to track one's finger precisely without line-of-sight enables a wide range of applications. However, we believe that the speed and precision which AuraBand's magnetic tracking enables unlocks additional functionality not commonly found in wearable sensing systems. In this section we discuss and demonstrate a few additional features that AuraBand can enable.

8.7.1 Tap Detection

Visions of free-hand interaction techniques in AR and VR commonly include in-air taps and gestures. However whether you are clicking in an app on AR glasses or drawing in a VR game, the sensation of touch is one of the critical factors for an immersive experience. The ability to robustly detect taps on environmental and body surfaces enables new kinds of always-available ambient interfaces [53, 186]. AuraBand helps enable this future by providing not only an estimate of finger position, but a high-speed robust signal which can be used to detect taps.

With a data rate of 472 Hz and a sensor bandwidth much higher than this, AuraBand is capable of measuring high speed events. To demonstrate this, Figure 8.15 shows a series of taps of varying intensities on a tabletop surface after a period of freely moving the finger and wrist in space. The lower portion of the figure shows a spectrogram with a window size of 128. While hand and finger motion contain mostly low frequency content, taps are immediately distinguishable due to their broadband spectrum and appear to contain content up to about 150 Hz.

8.7.2 Free-form and Subtle Hand Writing

To demonstrate that AuraBand is useful for much more than just gesture recognition, we show a few example traces of handwriting reconstruction using AuraBand. We consider two types of use cases. For use with a mixed reality HMD, we anticipate the use of wrist-tracking using fiducial markers embedded in the wristband. While one could use the wrist position as a virtual "pen", we argue that the dynamics of the wrist and finger joints are crucial to enabling a compelling experience. In Figure 8.16, we show two examples of writing the word "hello" in the air; one with big large movement over tens of centimeters (left) and another that was "written" on a notebook held in the opposing hand spanning just a couple of centimeters (right). The top row shows the result if one were to use just the wrist position; in this case, the wrist pose was captured from the motion capture system. On large motions, wrist position is a reasonable proxy that might be useful for gesture recognition, but not for precise input. For smaller motions the wrist position results in a poor quality signal. However, as shown in the bottom row, combining the estimate of finger position from AuraBand with a wrist estimate results in superior performance.

In other use cases, the wrist may not be tracked, such as with interaction on a smartwatch. In these cases, AuraBand is still capable of providing an accurate reconstruction of relative fingertip position, which is still useful for handwriting tasks. Figure 8.1 (left) shows an example of a handwriting trace reconstructed without any optical motion capture system. With appropriate visual feedback, we anticipate the performance to be even better.

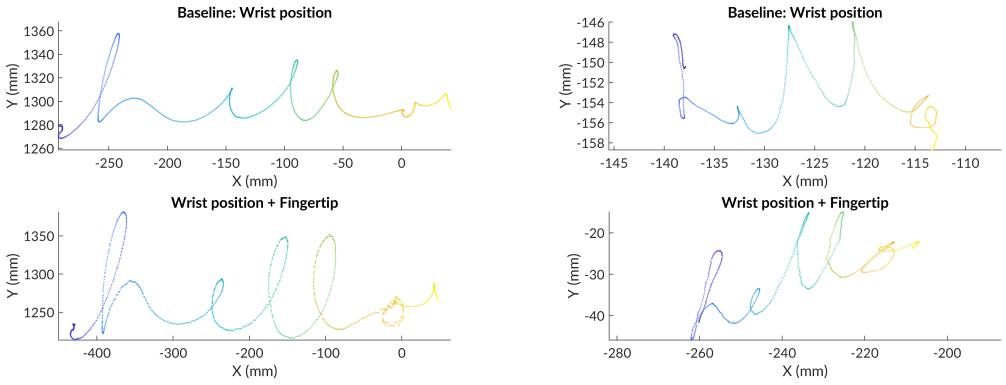

Figure 8.16: Handwriting examples using wrist-only motion frop motion capture (top) and wrist motion + AuraBand (bottom). The left example shows large hand motion. The right example show small writing on a handheld object.

8.8 Discussion

This chapter described AuraBand, a wearable device that enables precise tracking of the pose of a smart ring with respect to the wristband. The use of short-range electromagnetic tracking offers the unparalleled precision of EM tracking while minimizing the effects of environmental interference. In this work, we have proposed two methods for tracking. For more precise tracking we use an iterative model that achieves a position accuracy of $4.4\,\mathrm{mm}$ and a 2-DoF mean orientation accuracy of $4.65°$. In mobile scenarios where computation power is at a premium, we propose a relatively small neural network model with only one hidden layer. Using this approach AuraBand is able to achieve an accuracy of $6\,\mathrm{mm}$ across all users. We have also shown that for achieving even better accuracy, a per-session recalibration can reduce the position and orientation error to $1.5\,\mathrm{mm}$ and $1.7°$ respectively.

We envision AuraBand's use either with or without a head-mounted display. For mixed reality scenarios, head-mounted cameras are becoming increasingly capable of tracking the hands and fingers, but doing so comes at a significant power and compute cost. Tracking IR fiducial markers in a wristband is a much simpler task. However, in cases where full five-finger tracking is desired, AuraBand can serve as a robust prior to constrain the search space for a traditional hand-tracking algorithm. For interaction with other wearable devices, AuraBand can be used as an indirect input device.

8.8.1 Calibration Models

The use of a physics and sensor model is a significant advantage of our tracking algorithm, compared to other data-driven methods. It allows simulation and experimentation to optimize system configuration and decouples model training from user evaluation. That said, AuraBand does still require the use of a one-time factory calibration. For the prototype presented in this work, such a calibration is essential, due to the manufacturing tolerances of the device assembly. However, it is possible that for a mass-produced device, the tolerances can be made tight enough such that a single model would suffice for all units.

Although we intend for the factory calibration to be used across all sessions, we also presented results that showed that tweaking the calibration model using some per-session data can reduce the tracking error to $1.5\,\mathrm{mm}$. We attribute the change in calibration parameters between sessions to electrical effects in the signal processing chain. Several components in the pipeline, particularly the rectifier diodes, are sensitive to environmental temperature fluctuations. We expect that with more attention paid to analog stability, the global error will approach the per-session error. Another promising approach is migrating to digital processing for demodulation. This could enable reconstruction of both the magnitude and phase of the signals. We leave such explorations to future work.

8.8.2 Comparison to Related Work

Among related work in sensor-based tracking of the fingers, AuraBand is most similar to Finexus [21]. In AuraBand, we prioritized form-factor and designed the system from the ground up with wearability in mind—both in the design of the ring-worn transmitter and the wrist-based sensors. AuraBand uses significantly less power ($2.3\,\mathrm{mW}$ on the finger), uses an untethered ring, and estimates 2-DoF rotation in addition to position. In terms of accuracy, AuraBand's $1.5\,\mathrm{mm}$ single-session accuracy is comparable

to that reported in the single-session evaluation of Finexus. Moreover, the AuraBand accuracy holds at typical finger-wrist distances, which exceed the operating range of Finexus.

8.8.3 Extensions, Limitations, and Future Work

With some modifications, AuraBand can support tracking of multiple rings at the same time for full hand tracking. Currently the wristband is tuned to the transmitter frequency, but by tuning each ring to different frequencies, AuraBand can time multiplex between the ring's frequencies and measure the resulting fields. AuraBand has a maximum sensing range of $30\,\mathrm{cm}$. For multiple users in the same environment, there is likely little that needs to be done. However, if users wish to operate multiple devices in close proximity, different frequencies can be used.

AuraBand performance could further be improved by additional optimization of the sensor coil positions on the wristband. In general, the performance will improve as they are more spread out. Since we envision the ring to be worn on the index finger which is closer to the inside of the hand, we placed two sensors to the left side of the wristband, assuming right-handed use. AuraBand's hardware has the ability to support four sensor boards without modification of the electronics. For additional performance or for left-handed use, one could install another sensor board on the right side of the wristband.

A challenge in the design of any smart ring is sizing it for an appropriate fit across users. For the purposes of this study, we made the ring larger than average and used tape to wrap around the finger to ensure a snug fit. A more robust design might use several fixed size coils (e.g. small, medium, large) with replaceable insets in standard ring sizes.

Although this work focuses on the use of a ring to track the finger, AuraBand could also be used as a generic tracking platform for handheld objects. For example, by putting a coil around a stencil or pen, the object is now trackable with respect to the wrist. Pen input could be automatically digitized, a toy could be automatically turned into a gaming controller, or a toothbrush can monitor how someone is brushing their teeth.

Future work can also explore the use of a completely battery-free ring. The AuraBand system uses a transmitter on the ring and measures the resulting fields on the wristband. An alternative method of tracking could be to use AuraBand's wristband's coils as both the transmitter and receiver. In this topology, the ring is a tuned LC network that will change the measurements on the wristband based on its pose.

9 CONCLUSION

9.1 Toward an Ecosystem of Input Devices

In this dissertation, I present sensor-based approaches for input devices that are suitable for a range of next-generation computing platforms. Wearable form-factors, including mixed reality systems, are quickly becoming viable platforms for productivity, entertainment, and communication. For these platforms to realize their full potential, we need input devices that support the level of precise yet flexible input that these applications demand.

Importantly, different use cases demand different kinds of devices—there is no one-size-fits-all solution. For example, an engineer using mixed reality to design a 3D part needs a positionally tracked input device that does not require fatiguing arm motions. In contrast, a user responding to a message on public transportation needs an unobtrusive input device that enables rich communication. Just as today's input devices make use of mice, touchscreens, and voice, tomorrow's devices must also rely on a ecosystem of input devices to support diverse use cases. The devices optimized for mobile computing might take the form of wearable wristbands or rings while devices meant for long-term productivity might use eye tracking and lightweight controllers.

In Chapter 3, I described EyeContact, a new approach for high-precision scleral coil tracking [181]. Though not intended as a consumer device, this magnetic eye tracking approach can be used to prototype eye tracking solutions in HMDs, to calibrate other kinds of eye trackers, or to run perceptual studies that demand exceptional accuracy and precision. EyeContact models the behavior of magnetic fields generated around the HMD to bring a room-scale scleral coil tracker down to a device that simply snaps on to an existing HMD. This enables rapid experimentation without the need for a head-rest or bite-bar. EyeContact leverages both a data-driven approach for maximum performance and a model-driven approach for flexibility.

I then present a more consumer-friendly approach in Chapter 4 to achieve eye tracking using low-power photosensors. By embedding photosensors and infrared LEDs in a frame around the eye, we can capture reflections off of the sclera, iris, pupil, and skin. Based on the relative signals in each photosensor, we can estimate gaze. This approach has the advantage of significantly lower power and computation compared to camera-based trackers. I also present some initials steps toward tracking and compensating for shifts of the sensors or HMD on the face, a problem that notoriously plagues near-eye tracking systems.

Controllers represent today's dominant input device for HMDs, and this is likely to continue, as controllers offer a convenient platform for tracking and haptics. However, if such controllers are to be useful in mobile AR systems, they must be small enough to carry in a pocket when not in use. Today's controllers rely on optical tracking and feature a bulky tracking ring that limits their use outside of fixed environments. In Chapter 5, I present Aura, a novel 6-DoF electromagnetic tracking technique using a snap-on attachment for an HMD and a small form-factor controller. By measuring the magnetic fields emitting by three transmitter coils on the headset, the controller can estimate its 6-DoF pose relative to the head using

data-driven models. Such a system opens the design space for new form-factors of inside-out positional tracking systems that are more suitable for mobile use.

Handheld controllers also provide useful haptic feedback, but that feedback is generally limited to vibration. Vibrotactile actuators can provide a useful notification signal, but are fairly limited in the range of sensations they can generate. To explore a broader space of haptic feedback possibilities in virtual reality environments, I describe the Haptic Revolver in Chapter 6. This device uses a reconfigurable wheel embedded in the controller that raises and lowers to make contact with the finger as the virtual finger contacts virtual objects. The wheel spins to render shear force and directional cues against the fingertip and can rotate to place unique haptic elements on the wheel under the finger to match the virtual object being touched. This enables more realistic feedback and compelling interactive scenarios.

Although handheld tools are useful props for interacting with mixed reality environments, in other settings, we need finer-grained input using the fingers. Such finger-based input is useful for both small wearable device (e.g. a smartwatch) or larger head-mounted devices (e.g. Google Glass or HoloLens). A key aspect missing from these systems is the ability to productively input text. To that end, Chapter 7 presents DigiTouch, a glove-based device that tracks thumb-to-finger touches. This work explores the use of small thumb motions for interaction and text-entry. By using continuous tracking of the thumb along the fingers, DigiTouch enables reconfigurable input and adaptive text entry. The interaction techniques prototyped in DigiTouch are applicable to glove-based input or more generally to any hand-tracking system with precise thumb tracking.

Finally, Chapter 8 considers precise finger tracking using form-factors appropriate for everyday use. In this chapter, I discuss AuraBand, a wristband with embedded magnetic sensors that tracks the 5-DoF pose of a ring. With the ring worn on the index finger, this enables robust, precise finger tracking using both data-driven and model-driven algorithms. Such tracking can be used for drawing, handwriting tasks, or even animating a virtual hand using inverse kinematics models. Because the magnetic tracking is high-bandwidth, it can also detect taps on physical surfaces. By combining finger-wrist tracking with fiducial-based wrist-head tracking, AuraBand can estimate the absolute pose of the fingertip in head-space.

Each of these contributions push the boundary of the capabilities of sensor-based input devices. Taken together, these contributions represent small steps toward the kind of ecosystem we need to power our everyday computing platforms.

9.2 Reflections on Building Sensing Systems

This dissertation describes several examples of input devices with novel tracking, interaction, and haptic feedback. In this section, I offer a few reflections on useful strategies in designing input devices.

9.2.1 Continuous Tracking for Gesture Recognition

Interaction using hand or finger movement is often colloquially referred to as “gestures”. One might make a swipe gesture to scroll through a list, a tap gesture to select an item, or a unique hand gesture to control an application-specific command. Gesture-based interaction can be a useful and natural form of input

and should be supported by our input devices, including eye trackers, handheld controllers, and wearable finger tracking devices.

In building systems that support useful gestures, system designers should take care not to handicap the device by *only* supporting a fixed gesture set. An eye tracker is different from a device that detects blinks and saccades. A 6-DoF controller is different from a handheld device that recognizes when it is shaken or turned. Perhaps most salient for the HCI community, a device that tracks the finger is different from a device that detects finger movement in a particular direction Although a system that only supports gestures relies on signals that correlate with pose, a continuous tracking system must use those signals to estimate the absolute pose.

Of course, we should not build tracking systems out of principle alone. Instead, I argue that we should build input devices that strive for the highest fidelity reconstruction and support a wide range of interaction techniques. For example, a finger-tracking system like AuraBand supports pose estimation, but in practice, one would also build a gesture recognition system on top of the tracking signal. Separating the tracking from the interaction presents useful abstractions. For one, this allows the reuse of existing gesture recognition algorithms and existing gesture sets, without having to compromise based on the limitations of the sensing approach. Second, a gesture recognition system built atop a tracking system is fundamentally more robust, even with changes in user and session. Many systems that classify gestures directly from sensor signals require a user to recalibrate after putting on the device to retrain a model. If a system is tracking continuously though, it is likely more capable of detecting those differences between sessions and accounting for them. In AuraBand, for example, the tracking operates independent of how it is positioned on the wrist, but the system can automatically learn positioning of the band based on an inverse kinematics model. This allows reconstruction of pose and joint angles that serve as a more robust signal upon which to build a gesture set.

9.2.2 Parametric vs data-driven solutions

With advances in machine learning, particularly deep learning, there is renewed interest in "black box" models that encapsulate feature extraction and regression. These models can be used for gesture recognition, in the case of many wearable sensors, or for tracking (e.g. Aura). In contrast, parametric model-based approaches rely on first principles to construct a forward model that estimates the expected sensor signals given the variable of interest. Oftentimes data-driven approaches outperform parametric models based on first-principles. For example, in EyeContact, the data-driven neural network approach significantly outperformed a parametric physics model, simply because it was able to capture nuances in the data that are not part of the physics model. In other cases, such as in Aura, it is not possible to construct a parametric model due to the difficulty of modeling effects like field-distorting elements inside the HMD.

Nonetheless, in many cases it is beneficial to derive a model-driven solution, even if it cannot outperform a data-driven approach. Doing so forces the designer to confront the physics of the problem and often reveals insights that can improve either approach. For example, in Aura and AuraBand, constructing a parametric model of the analog processing helped explain the initial poor performance in tracking. By parameterizing the sensing in terms of the system noise, diode bias, and channel gains, we were able to model and explain our observations. Simply inverting this model, led to significant improvements in tracking quality. In AuraBand, the use of a model-driven tracking algorithm eliminates the need for any

user calibration. It allows the device to extrapolate to unseen data, while data-driven approaches would only be able to interpolate between data already seen. In other cases, developing such a model has led to new functionality. In DigiTouch, for example, modeling the contact resistance between the thumb and fingers enabled reconstruction of pressure in addition to touch location. Although it may not be possible in every situation, operating from first-principles can lend new insights into the problem.

Even model-driven approaches cannot operate in isolate from real-world data. For precise tracking, it is important to refine parameters of the model to more closely match the observed data. In EyeContact and AuraBand, this meant starting with the CAD-defined geometric positions of each coil, but learning slack parameters to fine-tune the position and orientation to match the measurements. Often this calibration step can take the form of a one-time factory calibration. In a manufacturing setting, one would need to carefully consider the tolerances of the coil position to determine whether such a calibration is necessary.

As a general principle, I recommend first deriving models of the phenomena and sensors if at all possible. This is useful for simulating the performance of a system, gaining a better understanding of the problem, and offers a principles-based tracking approach. When this isn't possible, or if the performance is insufficient, using data-driven approaches to calibrate the model is a good next step. Finally, in some cases, leveraging a completely data-driven approach can provide the best performance when one can ensure the training data covers all possible use cases.

9.2.3 On simulations and prototyping

Most of the projects presented in this dissertation rely on some form of simulation to inform the design of the device. In EyeContact, Aura, and AuraBand, magnetic field simulations allow the exploration of different configurations of the transmitters and receivers. In some cases, such as in EyeContact and AuraBand, the simulation eventually forms the basis for the forward model and tracking algorithm. In others, such as the rendering-based simulation for photosensor eye tracking, the simulation serves as an experimental platform to test hypotheses and optimize a potential design. Often, a simulation allows one to catch potentially fatal design flaws before investing time in hardware design.

When simulating a sensor system, one must make a decision about the fidelity of the simulation. Beyond the signal of interest, one must decide whether to include other factors like sensor shifts, user variation, and environmental noise. In some cases, such factors represent premature optimization and are best to simply measure on a hardware prototype. In other cases, including such parameters can reveal weaknesses that may limit the practical usability of the device.

As a practical example, in the original AuraBand simulation, we chose not to include factors like noise and slippage of the band on the wrist. The resulting simulation indicated that two coils along the wrist were sufficient to reconstruct finger pose. After bringing up a two-coil prototype, we quickly discovered that while the simulation held true in limited conditions, in real-world conditions where the band can wiggle slightly on the wrist, performance would degrade beyond a usable range. In this case, the simulation allowed us to easily verify that adding an additional coil would solve the problem, but had the simulation been higher fidelity, it would have saved prototyping cycles. There is never one right answer in deciding how far to go in simulation. Often, there are new insights or challenges that only arise after constructing a prototype. In my work throughout this thesis, I have generally adopted a synergistic approach in which the

simulation informs the prototype, but lessons learned from implementation are used to revise and refine the simulation.

9.2.4 Visualizing Real-time Signals

In sensor systems designed for interactive use, I have found it productive to implement real-time visualizations early in the design process. Visualizing live raw sensor data allows one to quickly catch mistakes or damaged hardware and better understand the strengths and weaknesses of the device. In initial prototypes of photodiode-based eye tracking, real-time visualization was critical to understand which sensors were most effective for different users. By adjusting the device on the face, one can determine sensor placements that optimize SNR. A higher-level visualization was particularly useful in Haptic Revolver, where a real-time 2D projection of the haptic wheel and nearby haptic elements enables rapid debugging and diagnosis.

Implementing such real-time visualizations is often tightly coupled to other real-time operations like filtering, inference, and logging. Often, implementing such tools can be time-consuming and error-prone, particularly for operation with high-speed signals. Through the course of this thesis work, I have developed a collection of tools in Python for real time data acquisition, processing, logging, and visualization. To enable others to more rapidly prototype real-time systems I have packaged these tools up as an open-source Python library, called PyRealtime [177]. This library uses a declarative dataflow-based syntax, in which the user specifies the data source and a number of operations to perform on the data and the library handles the implementation details. This effectively abstracts away the tedium of writing real-time multi-threaded code and allows the user to focus on signal processing and visualization.

9.3 Recommendations for Future Directions

The work in this dissertation made contributions in eye tracking, handheld controllers, and wearable devices. However, there is still much to be done to enable a productive ecosystem of input devices. In this section, I present some recommendations for promising avenues of future research in this domain.

9.3.1 Models for Sensor Fusion

Much of the work in this dissertation is motivated by a need for an ecosystem of input devices. For scoping purposes, the work here mostly focuses on each device or sensing technique separately—i.e. tracking a controller or tracking the fingers. However, significant benefits can be realized by intelligently combining different sources of information. For pure tracking systems, this is relatively straightforward. For example, in AuraBand, we proposed a optically-based head-wrist tracker and a magnetic-based wrist-finger tracker. With both sources of information, tracking the finger with respect to the head is a simple coordinate system transformation.

As our devices grow more sophisticated, multi-sensor integration becomes less straightforward. For example, head-mounted cameras might track the arm and wristband, which can move slightly on the wrist.

An IMU in the wristband might provide a coarse but high-speed pose estimate of the band. Using the techniques described in AuraBand, the band may also sense the pose of the finger, which may be supplemented by optical tracking data. The system may also have partial estimates of other aspects of body pose, like leg position based on movement of the entire body. One-off integrations of these different sensors using Kalman filters will likely prove unsustainable. Instead, I argue that we should work toward a full-body kinematic model and reframe each sensor as providing partial information about this model. Such a model allows both local and global optimization based on feasible poses.

At an operating system level, this framework provides a convenient abstraction through which to express other interaction techniques. For example, an application that allows the user to manipulate objects by pointing with a finger while the arm remains motionless at the side might request access to the finger pose with the respect to the wrist joint. Depending on the devices and sensors available on that user, such a request may or may not be feasible. In infeasible, the system may prompt the user to move to a pose compatible with the current sensor set.

9.3.2 Graceful Degradation

Such a sensor fusion framework also enables the concept of graceful degradation. Because our devices are used in so many contexts, it is important that they gracefully degrade as the situation grows more difficult. Consider a user who is using a mixed reality system to create precise 2D digital artwork with their fingers via direct manipulation. Suppose the fingers are tracked using a fusion of optical head-wrist tracking and magnetic wrist-finger tracking. To minimize fatigue, the user may wish to perform menu selection or some other task via indirect manipulation with the hands at the side, out of view of a head-mounted camera. In this situation, it is important that the interaction can gracefully degrade from absolute tracking and direct manipulation, to wrist-relative tracking and indirect manipulation.

Although sensor fusion models may provide a path forward for modeling uncertainty in tracking, it is also important to appropriately convey this uncertainty to the user through interface design. A user should not feel like they must relearn an entirely new interface whenever they operate the device in a slightly new way. Just as we need graceful degradation of tracking, we need graceful degradation of interfaces that are robust to changes in the context of use.

9.3.3 Online calibration

Many of the systems I present in this dissertation rely on some form of calibration. Some, like Aura and AuraBand are intended to use one-time factory calibrations Others, like EyeContact and photosensor-based eye tracking require per-session calibration. In a research prototype, a quick calibration session is likely acceptable, but for extensive use, we should explore options for online or implicit calibration. For eye tracking, self-calibrating approaches using saliency maps [158] or deep learning models of eye movement [174] are promising directions.

In an electromagnetic tracking system, such as Aura, the calibration may be invalidated at runtime if the system is operating near a large metal object in the environment. In such situations, one cannot simply ask the user to recalibrate. Instead, future research should explore sensor fusion techniques for online

calibration. Perhaps a head-mounted camera can track a fiducial on the controller that is only visible sometimes and only provides partial information about pose. By combining inertial measurements with partial pose estimates from a camera and a magnetic tracking, one could refine a magnetic calibration model and begin to model how the environment disrupts magnetic fields. Tracking data from external cameras, such as another HMD-wearing user would provide even more training data.

In summary

Our personal computing devices give us access to a wealth of information and tools that allow us to work, play, create, and communicate. As our devices grow more tightly coupled to our bodies, we must reduce the friction between us and our input devices. In wearable computing scenarios, we must work toward a set of devices that work together to sense intent from the user. In this dissertation, I have outlined on-head devices that enable eye tracking and around-body tracking, on-wrist devices that track the wrist and finger, and in-hand tools and accessories that give us new capabilities. Finally, we must consider an ecosystem of devices that work together to provide robust, precise, and subtle operation across an ever-increasing array of usage scenarios. It is my hope that others can build on the ideas in this dissertation to design intelligent sensor systems that increase our ability to be productive with our everyday devices.

BIBLIOGRAPHY

[1] 5DT. 5DT Data Glove Ultra Series. `http://www.5dt.com/downloads/dataglove/ultra/5DTDataGloveUltraDatasheet.pdf`. Accessed: 2019-05-09.

[2] AdHawk Microsystems. AdHawk Microsystems. `http://www.adhawkmicrosystems.com/`. Accessed: 2018-11-18.

[3] Sameer Agarwal, Keir Mierle, and Others. Ceres Solver. `http://ceres-solver.org`.

[4] Rachel Albert, Anjul Patney, David Luebke, and Joohwan Kim. Latency requirements for foveated rendering in virtual reality. *ACM Transactions on Applied Perception (TAP)*, 14(4):25, 2017.

[5] Christoph Amma, Marcus Georgi, and Tanja Schultz. Airwriting: Hands-free mobile text input by spotting and continuous recognition of 3d-space handwriting with inertial sensors. In *2012 16th International Symposium on Wearable Computers*, pages 52–59. IEEE, 2012.

[6] Bruno Araujo, Ricardo Jota, Varun Perumal, Jia Xian Yao, Karan Singh, and Daniel Wigdor. Snake Charmer: Physically enabling virtual objects. In *Proceedings of the TEI '16: Tenth International Conference on Tangible, Embedded, and Embodied Interaction*, TEI '16, pages 218–226, New York, NY, USA, 2016. ACM.

[7] Daniel Ashbrook, Patrick Baudisch, and Sean White. Nenya: subtle and eyes-free mobile input with a magnetically-tracked finger ring. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2043–2046. ACM, 2011.

[8] Mahdi Azmandian, Mark Hancock, Hrvoje Benko, Eyal Ofek, and Andrew D. Wilson. Haptic Retargeting: Dynamic repurposing of passive haptics for enhanced virtual reality experiences. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 1968–1979, New York, NY, USA, 2016. ACM.

[9] Teo Babic, Harald Reiterer, and Michael Haller. Pocket6: A 6DoF controller based on a simple smartphone application. In *SUI'18: 6th ACM Symposium on Spatial User Interaction*, pages 2–10, 2018.

[10] Yohan Baillot, L Davis, and J Rolland. A survey of tracking technology for virtual environments. *Fundamentals of wearable computers and augumented reality*, page 67, 2001.

[11] Daniel Barolet, François Christiaens, and Michael R Hamblin. Infrared and skin: Friend or foe. *Journal of Photochemistry and Photobiology B: Biology*, 155:78–85, 2016.

[12] Hrvoje Benko, Christian Holz, Mike Sinclair, and Eyal Ofek. NormalTouch and TextureTouch: High-fidelity 3d haptic shape rendering on handheld virtual reality controllers. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, UIST '16, pages 717–728, New York, NY, USA, 2016. ACM.

[13] Pascal Bérard, Derek Bradley, Markus Gross, and Thabo Beeler. Lightweight eye capture using a parametric model. *ACM Transactions on Graphics (TOG)*, 35(4):117, 2016.

[14] Mourad Bouzit, Grigore Burdea, George Popescu, and Rares Boian. The Rutgers Master II-new design force-feedback glove. *IEEE/ASME Transactions on mechatronics*, 7(2):256–263, 2002.

[15] Peter Bremen, Robert F Van der Willigen, and A John Van Opstal. Applying double magnetic induction to measure two-dimensional head-unrestrained gaze shifts in human subjects. *Journal of neurophysiology*, 98(6):3759–3769, 2007.

[16] Andreas Bulling, Daniel Roggen, and Gerhard Tröster. Wearable eog goggles: Seamless sensing and context-awareness in everyday environments. *Journal of Ambient Intelligence and Smart Environments*, 1(2):157–171, 2009.

[17] Tom Carter, Sue Ann Seah, Benjamin Long, Bruce Drinkwater, and Sriram Subramanian. UltraHaptics: Multi-point mid-air haptic feedback for touch surfaces. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, pages 505–514, New York, NY, USA, 2013. ACM.

[18] Liwei Chan, Yi-Ling Chen, Chi-Hao Hsieh, Rong-Hao Liang, and Bing-Yu Chen. CyclopsRing: Enabling whole-hand and context-aware interactions through a fisheye ring. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, pages 549–556. ACM, 2015.

[19] Liwei Chan, Rong-Hao Liang, Ming-Chang Tsai, Kai-Yin Cheng, Chao-Huai Su, Mike Y. Chen, Wen-Huang Cheng, and Bing-Yu Chen. FingerPad: Private and subtle interaction using fingertips. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, pages 255–260, New York, NY, USA, 2013. ACM.

[20] Ke-Yu Chen, Kent Lyons, Sean White, and Shwetak Patel. uTrack: 3d input using two magnetic sensors. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, pages 237–244, New York, NY, USA, 2013. ACM.

[21] Ke-Yu Chen, Shwetak N. Patel, and Sean Keller. Finexus: Tracking precise motions of multiple fingertips using magnetic sensing. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 1504–1514, New York, NY, USA, 2016. ACM.

[22] Wei-Tung Chen and Ling-Jyh Chen. Pokeball: A 3D positioning system using magnetism. In *Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2017 IEEE International Conference on*, pages 719–726. IEEE, 2017.

[23] Francesco Chinello, Monica Malvezzi, Claudio Pacchierotti, and Domenico Prattichizzo. Design and development of a 3rrs wearable fingertip cutaneous device. In *Advanced Intelligent Mechatronics (AIM), 2015 IEEE International Conference on*, pages 293–298. IEEE, 2015.

[24] Inrak Choi, Elliot W Hawkes, David L Christensen, Christopher J Ploch, and Sean Follmer. Wolverine: A wearable haptic interface for grasping in virtual reality. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 986–993. IEEE, 2016.

[25] Inrak Choi, Eyal Ofek, Hrvoje Benko, Mike Sinclair, and Christian Holz. CLAW: A multifunctional handheld haptic controller for grasping, touching, and triggering in virtual reality. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 654. ACM, 2018.

[26] Edward Clarkson, James Clawson, Kent Lyons, and Thad Starner. An empirical study of typing rates on mini-QWERTY keyboards. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '05, pages 1288–1291, New York, NY, USA, 2005. ACM.

[27] H Collewijn, F Van der Mark, and TC Jansen. Precise recording of human eye movements. *Vision research*, 15(3):447–IN5, 1975.

[28] B. Comet. An eye movement recording method operating with a closed eye. *Medical and Biological Engineering and Computing*, 21(5):628–631, Sep 1983.

[29] HTC Corporation. VIVE Pro Eye. `https://enterprise.vive.com/ca/product/vive-pro-eye/`. Accessed: 2019-06-06.

[30] H. Culbertson, J. Unwin, B. E. Goodman, and K. J. Kuchenbecker. Generating haptic texture models from unconstrained tool-surface interactions. In *2013 World Haptics Conference (WHC)*, pages 295–300, April 2013.

[31] CyberGlove Systems. CyberGrasp, 2017. Accessed: 2017-09-03.

[32] Houde Dai, Shuang Song, Chao Hu, Bo Sun, and Zhirong Lin. A novel 6D tracking method by fusion of 5D magnetic tracking and 3D inertial sensing. *IEEE Sensors Journal*, PP:1–1, 10 2018.

[33] Houde Dai, Shuang Song, Xianping Zeng, Shijian su, Mingqiang Lin, and Max Q.-H. Meng. 6D electromagnetic tracking approach using uniaxial transmitting coil and tri-axial magneto-resistive sensor. *IEEE Sensors Journal*, PP:1–1, 12 2017.

[34] Gisler Damian, Tobi Delbruck, and Patrick Lichtsteiner. Eye tracking using event-based silicon retina.

[35] Michael Deering. High resolution virtual reality. *SIGGRAPH Comput. Graph.*, 26(2):195–202, July 1992.

[36] Dexta Robotics. Dexmo, 2017. Accessed: 2017-09-03.

[37] David Dunn, Cary Tippets, Kent Torell, Petr Kellnhofer, Kaan Akşit, Piotr Didyk, Karol Myszkowski, David Luebke, and Henry Fuchs. Wide field of view varifocal near-eye display using see-through deformable membrane mirrors. *IEEE transactions on visualization and computer graphics*, 23(4):1322–1331, 2017.

[38] Augusto Esteves, Eduardo Velloso, Andreas Bulling, and Hans Gellersen. Orbits: Gaze interaction for smart watches using smooth pursuit eye movements. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, pages 457–466. ACM, 2015.

[39] exiii. EXOS Project, 2016. Accessed: 2017-09-03.

[40] FOVE. FOVE Eye Tracking Virtual Reality Headset. `https://www.getfove.com/`. Accessed: 2019-06-06.

[41] Eric Foxlin, Michael Harrington, and George Pfeifer. Constellation: A wide-range wireless motion-tracking system for augmented reality and virtual set applications. In *SIGGRAPH*, 1998.

[42] Marvin P Fried, Jonathan Kleefield, Harsha Gopal, Edward Reardon, Bryan T Ho, and Frederick A Kuhn. Image-guided endoscopic surgery: results of accuracy and performance in a multicenter clinical study using an electromagnetic tracking system. *The Laryngoscope*, 107(5):594–601, 1997.

[43] Maia Garau, Mel Slater, Vinoba Vinayagamoorthy, Andrea Brogni, Anthony Steed, and Martina Angela Sasse. The impact of avatar realism and eye gaze control on perceived quality of communication in a shared immersive virtual environment. In *Proceedings of the 2003 Conference on Human Factors in Computing Systems, CHI 2003, Ft. Lauderdale, Florida, USA, April 5-10, 2003*, pages 529–536, 2003.

[44] X. Ge, D. Lai, X. Wu, and Z. Fang. A novel non-model-based 6-dof electromagnetic tracking method using non-iterative algorithm. In *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 5144–5117, Sept 2009.

[45] Sarthak Ghosh, Hyeong Cheol Kim, Yang Cao, Arne Wessels, Simon T. Perrault, and Shengdong Zhao. Ringteraction: Coordinated thumb-index interaction using a ring. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '16, pages 2640–2647, New York, NY, USA, 2016. ACM.

[46] Peregrine Glove. Peregrine Glove ST. `https://peregrineglove.com/products/peregrine-glove-st-full-kit-w-pod`. Accessed: 2019-05-09.

[47] Go Touch VR. Touch the Virtual Reality with VR Touch, 2017. Accessed: 2017-09-06.

[48] Jun Gong, Yang Zhang, Xia Zhou, and Xing-Dong Yang. Pyro: Thumb-tip gesture recognition using pyroelectric infrared sensing. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, pages 553–563. ACM, 2017.

[49] Tovi Grossman, Xiang Anthony Chen, and George Fitzmaurice. Typing on glasses: Adapting text entry to smart eyewear. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '15, pages 144–152, New York, NY, USA, 2015. ACM.

[50] Brian Guenter, Mark Finch, Steven Drucker, Desney Tan, and John Snyder. Foveated 3D graphics. *ACM Transactions on Graphics (TOG)*, 31(6):164, 2012.

[51] Haption. Haption Virtuose 6D, 2013. Accessed: 2017-08-31.

[52] Chris Harrison and Scott E Hudson. Abracadabra: wireless, high-precision, and unpowered finger input for very small mobile devices. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pages 121–124. ACM, 2009.

[53] Chris Harrison, Desney Tan, and Dan Morris. Skinput: appropriating the body as an input surface. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 453–462. ACM, 2010.

[54] Florian Heller, Stefan Ivanov, Chat Wacharamanotham, and Jan Borchers. FabriTouch: Exploring flexible touch input on textiles. In *Proceedings of the 2014 ACM International Symposium on Wearable Computers*, ISWC '14, pages 59–62, New York, NY, USA, 2014. ACM.

[55] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. RGB-D mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *The International Journal of Robotics Research*, 31(5):647–663, 2012.

[56] Matthew Hirsch, Douglas Lanman, Henry Holtzman, and Ramesh Raskar. BiDi screen: a thin, depth-sensing lcd for 3d interaction using light fields. In *ACM Transactions on Graphics (ToG)*, volume 28, page 159. ACM, 2009.

[57] Christian Holz and Patrick Baudisch. Understanding touch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 2501–2510, New York, NY, USA, 2011. ACM.

[58] Mark MJ Houben, Janine Goumans, and Johannes van der Steen. Recording three-dimensional eye movements: scleral search coils versus video oculography. *Investigative Ophthalmology & Visual Science*, 47(1):179–187, 2006.

[59] Yi-Ta Hsieh, Antti Jylhä, Valeria Orso, Luciano Gamberini, and Giulio Jacucci. Designing a willing-to-use-in-public hand gestural interaction technique for smart glasses. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 4203–4215, New York, NY, USA, 2016. ACM.

[60] Xinda Hu and Hong Hua. High-resolution optical see-through multi-focal-plane head-mounted display using freeform optics. *Optics express*, 22(11):13896–13903, 2014.

[61] Da-Yuan Huang, Liwei Chan, Shuo Yang, Fan Wang, Rong-Hao Liang, De-Nian Yang, Yi-Ping Hung, and Bing-Yu Chen. DigitSpace: Designing thumb-to-fingers touch interfaces for one-handed and eyes-free interactions. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 1526–1537. ACM, 2016.

[62] SensoMotoric Instruments. Eye Tracking HMD Upgrade Package for Oculus Rift DK2. `http://www.smivision.com/en/gaze-and-eye-tracking-systems/products/eye-tracking-hmd-upgrade.html`. Accessed: 2015-05-26.

[63] Kenji Irie, Bruce A Wilson, Richard D Jones, Philip J Bones, and Tim J Anderson. A laser-based eye-tracking system. *Behavior Research Methods, Instruments, & Computers*, 34(4):561–572, 2002.

[64] Yoshio Ishiguro, Adiyan Mujibiya, Takashi Miyaki, and Jun Rekimoto. Aided Eyes: Eye activity sensing for daily life. In *Proceedings of the 1st Augmented Human International Conference*, AH '10, pages 25:1–25:7, New York, NY, USA, 2010. ACM.

[65] Mohd Noor Islam and Andrew J Fleming. Resonance-enhanced coupling for range extension of electromagnetic tracking systems. *IEEE Transactions on Magnetics*, 54(4):1–9, 2018.

[66] Hsin-Liu (Cindy) Kao, Artem Dementyev, Joseph A. Paradiso, and Chris Schmandt. NailO: Fingernails as an input surface. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 3015–3018, New York, NY, USA, 2015. ACM.

[67] Moritz Kassner, William Patera, and Andreas Bulling. Pupil: an open source platform for pervasive eye tracking and mobile gaze-based interaction. In *Proceedings of the 2014 ACM international joint conference on pervasive and ubiquitous computing: Adjunct publication*, pages 1151–1160. ACM, 2014.

[68] Dmytro Katrychuk, Henry Griffith, and Oleg Komogortsev. Power-efficient and shift-robust eye-tracking sensor for portable vr headsets. 2019.

[69] Wolf Kienzle and Ken Hinckley. LightRing: Always-available 2d input on any surface. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, pages 157–160, New York, NY, USA, 2014. ACM.

[70] David Kim, Otmar Hilliges, Shahram Izadi, Alex D. Butler, Jiawen Chen, Iason Oikonomidis, and Patrick Olivier. Digits: Freehand 3d interactions anywhere using a wrist-worn gloveless sensor. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '12, pages 167–176, New York, NY, USA, 2012. ACM.

[71] Joohwan Kim, Michael Stengel, Alexander Majercik, Shalini De Mello, David Dunn, Samuli Laine, Morgan McGuire, and David Luebke. NVGaze: An anatomically-informed dataset for low-latency, near-eye gaze estimation. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, page 550. ACM, 2019.

[72] Kyung-Nam Kim and RS Ramakrishna. Vision-based eye-gaze tracking for human computer interface. In *Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on*, volume 2, pages 324–329. IEEE, 1999.

[73] Wooyoung Kim, Jihoon Song, and Frank C Park. Closed-form position and orientation estimation for a three-axis electromagnetic tracking system. *IEEE Transactions on Industrial Electronics*, 65(5):4331–4337, 2018.

[74] Volodymyr V Kindratenko. A survey of electromagnetic position tracker calibration techniques. *Virtual Reality*, 5(3):169–182, 2000.

[75] Volodymyr V Kindratenko and William R Sherman. Neural network-based calibration of electromagnetic tracking systems. *Virtual Reality*, 9(1):70–78, 2005.

[76] Marion Koelle, Matthias Kranz, and Andreas Möller. Don't look at me that way!: Understanding user attitudes towards data glasses usage. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '15, pages 362–372, New York, NY, USA, 2015. ACM.

[77] Diana Krusteva, Deepak Sahoo, Asier Marzo, Sriram Subramanian, and David Coyle. Marionette: A multi-finger tilt feedback device for curvatures and haptic images perception. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '15, pages 1229–1234, New York, NY, USA, 2015. ACM.

[78] Katherine J Kuchenbecker, David Ferguson, Michael Kutzer, Matthew Moses, and Allison M Okamura. The touch thimble: Providing fingertip contact feedback during point-force haptic interaction. In *Haptic interfaces for virtual environment and teleoperator systems, 2008. haptics 2008. symposium on*, pages 239–246. IEEE, 2008.

[79] Falko Kuester, Michelle Chen, Mark E. Phair, and Carsten Mehring. Towards keyboard independent touch typing in VR. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, VRST '05, pages 86–95, New York, NY, USA, 2005. ACM.

[80] Jack B Kuipers. SPASYN-an electromagnetic relative position and orientation tracking system. *IEEE Transactions on Instrumentation and Measurement*, 29(4):462–466, 1980.

[81] Virtual Motion Labs. Vmg 30 plus. `http://www.virtualmotionlabs.com/products/vmg-30-plus/`. Accessed: 2016-03-14.

[82] Katja M Langen, Twyla R Willoughby, Sanford L Meeks, Anand Santhanam, Alexis Cunningham, Lisa Levine, and Patrick A Kupelian. Observations on real-time prostate gland motion using electromagnetic tracking. *International Journal of Radiation Oncology* Biology* Physics*, 71(4):1084–1090, 2008.

[83] Magic Leap. Magic Leap Fact Sheet. `https://www.magicleap.com/static/magic-leap-fact-sheet.pdf`. Accessed: 2019-04-05.

[84] Jaeyeon Lee, Mike Sinclair, Mar Gonzalez-Franco, Eyal Ofek, and Christian Holz. TORC: A virtual reality controller for in-hand high-dexterity finger interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, page 71. ACM, 2019.

[85] Seongil Lee, Sang Hyuk Hong, and Jae Wook Jeon. Designing a universal keyboard using chording gloves. In *Proceedings of the 2003 Conference on Universal Usability*, CUU '03, pages 142–147, New York, NY, USA, 2003. ACM.

[86] Tianxing Li, Qiang Liu, and Xia Zhou. Ultra-low power gaze tracking for virtual reality. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, page 25. ACM, 2017.

[87] Tianxing Li and Xia Zhou. Battery-free eye tracker on glasses. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, pages 67–82. ACM, 2018.

[88] Mingyu Liu, Mathieu Nancel, and Daniel Vogel. Gunslinger: Subtle arms-down mid-air interaction. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, pages 63–71. ACM, 2015.

[89] Kent Lyons. 2D input for virtual reality enclosures with magnetic field sensing. In *Proceedings of the 2016 ACM International Symposium on Wearable Computers*, pages 176–183. ACM, 2016.

[90] Kent Lyons, Thad Starner, Daniel Plaisted, James Fusia, Amanda Lyons, Aaron Drew, and E. W. Looney. Twiddler typing: One-handed chording text entry for mobile phones. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 671–678, New York, NY, USA, 2004. ACM.

[91] I. Scott MacKenzie and R. William Soukoreff. A character-level error analysis technique for evaluating text entry methods. In *Proceedings of the Second Nordic Conference on Human-computer Interaction*, NordiCHI '02, pages 243–246, New York, NY, USA, 2002. ACM.

[92] I. Scott MacKenzie and R. William Soukoreff. Text entry for mobile computing: Models and methods,theory and practice. *Human-Computer Interaction*, 17(2-3):147–198, 2002.

[93] I. Scott MacKenzie and R. William Soukoreff. Phrase sets for evaluating text entry techniques. In *CHI '03 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '03, pages 754–755, New York, NY, USA, 2003. ACM.

[94] F Landis Markley and Daniele Mortari. How to estimate attitude from vector observations. 1999.

[95] Anders Markussen, Mikkel Rønne Jakobsen, and Kasper Hornbæk. Vulture: A mid-air word-gesture keyboard. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14, pages 1073–1082, New York, NY, USA, 2014. ACM.

[96] Susana Martinez-Conde, Jorge Otero-Millan, and Stephen L Macknik. The impact of microsaccades on vision: towards a unified theory of saccadic function. *Nature Reviews Neuroscience*, 14(2):83–96, 2013.

[97] Thomas H Massie, J Kenneth Salisbury, et al. The phantom haptic interface: A device for probing virtual objects. In *Proceedings of the ASME winter annual meeting, symposium on haptic interfaces for virtual environment and teleoperator systems*, volume 55, pages 295–300. Chicago, IL, 1994.

[98] Nathan Matsuda, Alexander Fix, and Douglas Lanman. Focal surface displays. *ACM Transactions on Graphics (TOG)*, 36(4):86, 2017.

[99] Addison Mayberry, Pan Hu, Benjamin Marlin, Christopher Salthouse, and Deepak Ganesan. iShadow: Design of a wearable, real-time mobile gaze tracker. In *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '14, pages 82–94, New York, NY, USA, 2014. ACM.

[100] Addison Mayberry, Yamin Tun, Pan Hu, Duncan Smith-Freedman, Deepak Ganesan, Benjamin M. Marlin, and Christopher Salthouse. CIDER: Enabling robustness-power tradeoffs on a computational eyeglass. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, MobiCom '15, pages 400–412, New York, NY, USA, 2015. ACM.

[101] Michael Meehan, Brent Insko, Mary Whitton, and Frederick P Brooks Jr. Physiological measures of presence in stressful virtual environments. *ACM Transactions on Graphics (TOG)*, 21(3):645–652, 2002.

[102] CGM Meskers, HM Vermeulen, JH De Groot, FCT Van der Helm, and PM Rozing. 3d shoulder position measurements using a six-degree-of-freedom electromagnetic tracking device. *Clinical biomechanics*, 13(4):280–292, 1998.

[103] Sam Miller, Andy Smith, Sina Bahram, and Robert St. Amant. A glove for tapping and discrete 1d/2d input. In *Proceedings of the 2012 ACM International Conference on Intelligent User Interfaces*, IUI '12, pages 101–104, New York, NY, USA, 2012. ACM.

[104] Dan Morris, Hong Tan, Federico Barbagli, Timothy Chang, and Kenneth Salisbury. Haptic feedback enhances force skill learning. In *EuroHaptics Conference, 2007 and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems. World Haptics 2007. Second Joint*, pages 21–26. IEEE, 2007.

[105] Suranga Nanayakkara, Roy Shilkrot, Kian Peen Yeo, and Pattie Maes. EyeRing: a finger-worn input device for seamless interactions with our surroundings. In *Proceedings of the 4th Augmented Human International Conference*, pages 13–20. ACM, 2013.

[106] Rajalakshmi Nandakumar, Vikram Iyer, and Shyamnath Gollakota. 3D localization for sub-

centimeter sized devices. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, pages 108–119. ACM, 2018.

[107] Rajalakshmi Nandakumar, Vikram Iyer, Desney Tan, and Shyamnath Gollakota. FingerIO: Using active sonar for fine-grained finger tracking. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 1515–1525. ACM, 2016.

[108] NDI. Medical Aurora - Medical. `https://www.ndigital.com/medical/products/aurora/`. Accessed: 2019-05-09.

[109] Novint. Novint Falcon, 2011. Accessed: 2017-09-03.

[110] Claudio Pacchierotti, Domenico Prattichizzo, and Katherine J Kuchenbecker. Displaying sensed tactile cues with a fingertip haptic device. *IEEE transactions on haptics*, 8(4):384–396, 2015.

[111] Claudio Pacchierotti, Gionata Salvietti, Irfan Hussain, Leonardo Meli, and Domenico Prattichizzo. The hRing: A wearable haptic device to avoid occlusions in hand tracking. In *Haptics Symposium (HAPTICS), 2016 IEEE*, pages 134–139. IEEE, 2016.

[112] Claudio Pacchierotti, Stephen Sinclair, Massimiliano Solazzi, Antonio Frisoli, Vincent Hayward, and Domenico Prattichizzo. Wearable haptic systems for the fingertip and the hand: taxonomy, review, and perspectives. *IEEE Transactions on Haptics*, 2017.

[113] Rohit Pandey, Pavel Pidlypenskyi, Shuoran Yang, and Christine Kaeser-Chen. Egocentric 6-dof tracking of small handheld objects. *CoRR*, abs/1804.05870, 2018.

[114] E. Paperno, I. Sasada, and E. Leonovich. A new method for magnetic position and orientation tracking. *IEEE Transactions on Magnetics*, 37:1938–1940, July 2001.

[115] Kurt Partridge, Saurav Chatterjee, Vibha Sazawal, Gaetano Borriello, and Roy Want. TiltType: Accelerometer-supported text entry for very small devices. In *Proceedings of the 15th Annual ACM Symposium on User Interface Software and Technology*, UIST '02, pages 201–204, New York, NY, USA, 2002. ACM.

[116] Valter Pasku, Alessio De Angelis, Guido De Angelis, Darmindra D Arumugam, Marco Dionigi, Paolo Carbone, Antonio Moschitta, and David S Ricketts. Magnetic field-based positioning systems. *IEEE Communications Surveys & Tutorials*, 19(3):2003–2017, 2017.

[117] Alvaro G Perez, Daniel Lobo, Francesco Chinello, Gabriel Cirio, Monica Malvezzi, José San Martín, Domenico Prattichizzo, and Miguel A Otaduy. Soft finger tactile rendering for wearable haptics. In *World Haptics Conference (WHC), 2015 IEEE*, pages 327–332. IEEE, 2015.

[118] D Perie, AJ Tate, PL Cheng, and GA Dumas. Evaluation and calibration of an electromagnetic tracking device for biomechanical analysis of lifting tasks. *Journal of biomechanics*, 35(2):293–297, 2002.

[119] Anna Peshock, Julia Duvall, and Lucy E. Dunne. Argot: A wearable one-handed keyboard glove. In *Proceedings of the 2014 ACM International Symposium on Wearable Computers: Adjunct Program*, ISWC '14 Adjunct, pages 87–92, New York, NY, USA, 2014. ACM.

[120] Ken Pfeuffer, Jason Alexander, Ming Ki Chong, Yanxia Zhang, and Hans Gellersen. Gaze-shifting:

Direct-indirect input with pen and touch modulated by gaze. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, pages 373–383. ACM, 2015.

[121] Ken Pfeuffer, Melodie Vidal, Jayson Turner, Andreas Bulling, and Hans Gellersen. Pursuit calibration: Making gaze calibration less tedious and more flexible. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*, pages 261–270. ACM, 2013.

[122] G. Pirkl, K. Stockinger, K. Kunze, and P. Lukowicz. Adapting magnetic resonant coupling based relative positioning technology for wearable activitiy recogniton. In *2008 12th IEEE International Symposium on Wearable Computers*, pages 47–54, Sept 2008.

[123] Anton Plotkin and Eugene Paperno. 3-D magnetic tracking of a single subminiature coil with a large 2-d array of uniaxial transmitters. *IEEE Transactions on Magnetics*, 39(5):3295–3297, 2003.

[124] Anton Plotkin, Oren Shafrir, Eugene Paperno, and Daniel M Kaplan. Magnetic eye tracking: A new approach employing a planar transmitter. *Biomedical Engineering, IEEE Transactions on*, 57(5):1209–1215, 2010.

[125] Polhemus. Polhemus G4. `https://polhemus.com/motion-tracking/all-trackers/g4`. Accessed: 2018-12-11.

[126] Ivan Poupyrev, Nan-Wei Gong, Shiho Fukuhara, Mustafa Emre Karagozler, Carsten Schwesig, and Karen E. Robinson. Project Jacquard: Interactive digital textiles at scale. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 4216–4227, New York, NY, USA, 2016. ACM.

[127] Manuel Pratorius, Ulrich Burgbacher, Dimitar Valkov, and Klaus Hinrichs. Sensing thumb-to-finger taps for symbolic input in vr/ar environments. *IEEE computer graphics and applications*, (5):42–54, 2015.

[128] Domenico Prattichizzo, Francesco Chinello, Claudio Pacchierotti, and Monica Malvezzi. Towards wearability in fingertip haptics: a 3-dof wearable device for cutaneous force feedback. *IEEE Transactions on Haptics*, 6(4):506–516, 2013.

[129] William R Provancher. Creating greater VR immersion by emulating force feedback with ungrounded tactile feedback. *IQT Quarterly*, 6(2):18–21, 2014.

[130] L. Quéval. BSmag toolbox user manual. Technical report, Dept. Elect. Eng., University of Applied Sciences Düsseldorf, Düsseldorf, Germany, April 2015. Accessed: 2018-12-11.

[131] Frederick H Raab, Ernest B Blood, Terry O Steiner, and Herbert R Jones. Magnetic position and orientation tracking system. *IEEE Transactions on Aerospace and Electronic systems*, (5):709–718, 1979.

[132] SR Research. EyeLink 1000 Plus. `https://www.sr-research.com/products/eyelink-1000-plus/`. Accessed: 2019-06-06.

[133] JPH Reulen and L Bakker. The measurement of eye movement using double magnetic induction. *Biomedical Engineering, IEEE Transactions on*, (11):740–744, 1982.

[134] Gabriel Reyes, Jason Wu, Nikita Juneja, Maxim Goldshtein, W. Keith Edwards, Gregory D. Abowd, and Thad Starner. SynchroWatch: One-handed synchronous smartwatch gestures using correlation and magnetic sensing. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 1(4):158:1–158:26, January 2018.

[135] Ioannis Rigas, Hayes Raffle, and Oleg V Komogortsev. Hybrid PS-V Technique: A novel sensor fusion approach for fast mobile eye-tracking with sensor-shift aware correction. *IEEE Sensors Journal*, 17(24):8356–8366, 2017.

[136] Ioannis Rigas, Hayes Raffle, and Oleg V Komogortsev. Photosensor oculography: Survey and parametric analysis of designs using model-based simulation. *IEEE Transactions on Human-Machine Systems*, (99):1–12, 2018.

[137] Dale Roberts, Mark Shelhamer, and Aaron Wong. A new wireless search-coil system. In *Proceedings of the 2008 symposium on Eye tracking research & applications*, pages 197–204. ACM, 2008.

[138] David Robinson et al. A method of measuring eye movemnent using a scieral search coil in a magnetic field. *Bio-medical Electronics, IEEE Transactions on*, 10(4):137–145, 1963.

[139] D. Roetenberg, P. J. Slycke, and P. H. Veltink. Ambulatory position and orientation tracking fusing magnetic and inertial sensing. *IEEE Transactions on Biomedical Engineering*, 54(5):883–890, May 2007.

[140] R. Rosenberg and M. Slater. The Chording Glove: A glove-based text input device. *Trans. Sys. Man Cyber Part C*, 29(2):186–191, May 1999.

[141] Soha Rostaminia, Addison Mayberry, Deepak Ganesan, Benjamin Marlin, and Jeremy Gummeson. iLid: Low-power sensing of fatigue and drowsiness measures on a computational eyeglass. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 1(2):23:1–23:26, June 2017.

[142] J Edward Russo. The limbus reflection method for measuring eye position. *Behavior Research Methods & Instrumentation*, 7(2):205–208, 1975.

[143] T. Scott Saponas, Desney S. Tan, Dan Morris, Ravin Balakrishnan, Jim Turner, and James A. Landay. Enabling always-available input with muscle-computer interfaces. In *Proceedings of the 22Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '09, pages 167–176, New York, NY, USA, 2009. ACM.

[144] Stefan Schneegass and Alexandra Voit. GestureSleeve: Using touch sensitive fabrics for gestural input on the forearm for controlling smartwatches. In *Proceedings of the 2016 ACM International Symposium on Wearable Computers*, ISWC '16, pages 108–115, New York, NY, USA, 2016. ACM.

[145] Samuel B. Schorr and Allison M. Okamura. Fingertip tactile devices for virtual object manipulation and exploration. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 3115–3119, New York, NY, USA, 2017. ACM.

[146] Sheng Shen, He Wang, and Romit Roy Choudhury. I am a smartwatch and I can track my user's arm. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '16, pages 85–96, New York, NY, USA, 2016. ACM.

[147] Massimiliano Solazzi, Antonio Frisoli, and Massimo Bergamasco. Design of a novel finger haptic interface for contact and orientation display. In *Haptics Symposium, 2010 IEEE*, pages 129–132. IEEE, 2010.

[148] S. Song, W. Qiao, B. Li, C. Hu, H. Ren, and M. Q. . Meng. An efficient magnetic tracking method using uniaxial sensing coil. *IEEE Transactions on Magnetics*, 50(1):1–7, Jan 2014.

[149] Shuang Song, Chao Hu, Baopu Li, Xiaoxiao Li, and Max Q.-H. Meng. An electromagnetic localization and orientation method based on rotating magnetic dipole. *IEEE Transactions on Magnetics*, 49:1274–1277, 03 2013.

[150] Shuang Song, Hongliang Ren, and Haoyong Yu. An improved magnetic tracking method using rotating uniaxial coil with sparse points and closed form analytic solution. *IEEE Sensors Journal*, 14:3585–3592, 2014.

[151] Olga Sorkine-Hornung and Michael Rabinovich. Least-squares rigid motion using svd. 2017.

[152] R. William Soukoreff and I. Scott MacKenzie. Metrics for text entry research: An evaluation of MSD and KSPC, and a new unified error metric. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, pages 113–120, New York, NY, USA, 2003. ACM.

[153] Srinath Sridhar, Anna Maria Feit, Christian Theobalt, and Antti Oulasvirta. Investigating the dexterity of multi-finger input for mid-air text entry. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 3643–3652, New York, NY, USA, 2015. ACM.

[154] Thomas Stiefmeier, Georg Ogris, Holger Junker, Paul Lukowicz, and Gerhard Troster. Combining motion sensors and ultrasonic hands tracking for continuous activity recognition in a maintenance scenario. In *Wearable computers, 2006 10th IEEE international symposium on*, pages 97–104. IEEE, 2006.

[155] Evan Strasnick, Christian Holz, Eyal Ofek, Mike Sinclair, and Hrvoje Benko. Haptic Links: Bimanual haptics for virtual reality using variable stiffness actuation. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 644. ACM, 2018.

[156] Paul Strohmeier and Kasper Hornbæk. Generating haptic textures with a vibrotactile actuator. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 4994–5005, New York, NY, USA, 2017. ACM.

[157] D. J. Sturman and D. Zeltzer. A survey of glove-based input. *IEEE Computer Graphics and Applications*, 14(1):30–39, Jan 1994.

[158] Yusuke Sugano, Yasuyuki Matsushita, and Yoichi Sato. Calibration-free gaze sensing using saliency maps. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2667–2674. IEEE, 2010.

[159] CyberGlove Systems. CyberGlove II. `http://www.cyberglovesystems.com/cyberglove-ii`. Accessed: 2019-05-09.

[160] Jonathan Taylor, Lucas Bordeaux, Thomas Cashman, Bob Corish, Cem Keskin, Toby Sharp, Eduardo Soto, David Sweeney, Julien Valentin, Benjamin Luff, et al. Efficient and precise interactive hand

tracking through joint, continuous optimization of pose and correspondences. *ACM Transactions on Graphics (TOG)*, 35(4):143, 2016.

[161] Lore Thaler, Alexander C Schütz, Melvyn A Goodale, and Karl R Gegenfurtner. What is the best fixation target? the effect of target shape on stability of fixational eye movements. *Vision Research*, 76:31–42, 2013.

[162] Jakob S Thomassen, Giacomo Di Benedetto, and Bernhard JM Hess. Decoding 3d search coil signals in a non-homogeneous magnetic field. *Vision research*, 50(13):1203–1213, 2010.

[163] Marc Tonsen, Julian Steil, Yusuke Sugano, and Andreas Bulling. InvisibleEye: Mobile eye tracking using multiple low-resolution cameras and learning-based gaze estimation. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 1(3):106:1–106:21, September 2017.

[164] C. Topal, A. Dogan, and O. N. Gerek. A wearable head-mounted sensor-based apparatus for eye tracking applications. In *2008 IEEE Conference on Virtual Environments, Human-Computer Interfaces and Measurement Systems*, pages 136–139, July 2008.

[165] C. Topal, S. Gunal, O. KoÃğdeviren, A. DoÄ§an, and ÃŰ. N. Gerek. A low-computational approach on gaze estimation with eye touch system. *IEEE Transactions on Cybernetics*, 44(2):228–239, Feb 2014.

[166] Cihan Topal, Ömer Nezih Gerek, and Atakan Doğan. A head-mounted sensor-based eye tracking device: Eye Touch System. In *Proceedings of the 2008 Symposium on Eye Tracking Research & Applications*, ETRA '08, pages 87–90, New York, NY, USA, 2008. ACM.

[167] Ying-Chao Tung, Chun-Yen Hsu, Han-Yu Wang, Silvia Chyou, Jhe-Wei Lin, Pei-Jung Wu, Andries Valstar, and Mike Y. Chen. User-defined game input for smart glasses in public space. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 3327–3336, New York, NY, USA, 2015. ACM.

[168] Richard Q Van der Linde, Piet Lammertse, Erwin Frederiksen, and B Ruiter. The HapticMaster, a new high-performance haptic interface. In *Proc. Eurohaptics*, pages 1–5, 2002.

[169] Olivier AJ Van der Meijden and Marlies P Schijven. The value of haptic feedback in conventional and robot-assisted minimal invasive surgery and virtual reality training: a current review. *Surgical endoscopy*, 23(6):1180–1190, 2009.

[170] Varjo. Varjo Bionic Display. `https://varjo.com/bionic-display/`. Accessed: 2018-11-19.

[171] Bill Verplank, Michael Gurevich, and Max Mathews. The Plank: Designing a simple haptic controller. In *Proceedings of the 2002 Conference on New Interfaces for Musical Expression*, NIME '02, pages 1–4, Singapore, Singapore, 2002. National University of Singapore.

[172] Anran Wang and Shyamnath Gollakota. MilliSonic: Pushing the limits of acoustic motion tracking. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, pages 18:1–18:11, New York, NY, USA, 2019. ACM.

[173] Cheng-Yao Wang, Wei-Chen Chu, Po-Tsung Chiu, Min-Chieh Hsiu, Yih-Harn Chiang, and Mike Y. Chen. PalmType: Using palms as keyboards for smart glasses. In *Proceedings of the 17th International*

Conference on Human-Computer Interaction with Mobile Devices and Services, MobileHCI '15, pages 153–160, New York, NY, USA, 2015. ACM.

[174] Kang Wang, Shen Wang, and Qiang Ji. Deep eye fixation map learning for calibration-free eye gaze tracking. In *Proceedings of the ninth biennial ACM symposium on eye tracking research & applications*, pages 47–55. ACM, 2016.

[175] Robert Y. Wang and Jovan Popović. Real-time hand-tracking with a color glove. In *ACM SIGGRAPH 2009 Papers*, SIGGRAPH '09, pages 63:1–63:8, New York, NY, USA, 2009. ACM.

[176] Saiwen Wang, Jie Song, Jaime Lien, Ivan Poupyrev, and Otmar Hilliges. Interacting with Soli: Exploring fine-grained dynamic gesture recognition in the radio-frequency spectrum. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pages 851–860. ACM, 2016.

[177] Eric Whitmire. PyRealtime. `https://github.com/ewhitmire/pyrealtime`, 2017.

[178] Eric Whitmire, Hrvoje Benko, Christian Holz, Eyal Ofek, and Mike Sinclair. Haptic Revolver: Touch, shear, texture, and shape rendering on a reconfigurable virtual reality controller. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 86. ACM, 2018.

[179] Eric Whitmire, Mohit Jain, Divye Jain, Greg Nelson, Ravi Karkar, Shwetak Patel, and Mayank Goel. DigiTouch: Reconfigurable thumb-to-finger input and text entry on head-mounted displays. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(3):113, 2017.

[180] Eric Whitmire, Farshid Salemi Parizi, and Shwetak Patel. Aura: Inside-out electromagnetic controller tracking. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 2019.

[181] Eric Whitmire, Laura Trutoiu, Robert Cavin, David Perek, Brian Scally, James Phillips, and Shwetak Patel. EyeContact: scleral coil eye tracking for virtual reality. In *Proceedings of the 2016 ACM International Symposium on Wearable Computers*, pages 184–191. ACM, 2016.

[182] Maarten WA Wijntjes, Akihiro Sato, Vincent Hayward, and Astrid ML Kappers. Local surface orientation dominates haptic curvature discrimination. *IEEE transactions on haptics*, 2(2):94–102, 2009.

[183] Jacob O. Wobbrock and Brad A. Myers. Analyzing the input stream for character- level errors in unconstrained text entry evaluations. *ACM Trans. Comput.-Hum. Interact.*, 13(4):458–489, December 2006.

[184] Haijun Xia, Tovi Grossman, and George Fitzmaurice. NanoStylus: Enhancing input on ultra-small displays with a finger-mounted stylus. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, pages 447–456. ACM, 2015.

[185] Guofang Xiao, Ester Bonmati, Stephen Thompson, Joe Evans, John Hipwell, Daniil Nikitichev, Kurinchi Gurusamy, Sébastien Ourselin, David J Hawkes, Brian Davidson, et al. Electromagnetic tracking in image-guided laparoscopic surgery: Comparison with optical tracking and feasibility study of a combined laparoscope and laparoscopic ultrasound system. *Medical physics*, 45(11):5094–5104, 2018.

[186] Robert Xiao, Julia Schwarz, Nick Throm, Andrew D Wilson, and Hrvoje Benko. Mrtouch: Adding

touch input to head-mounted mixed reality. *IEEE transactions on visualization and computer graphics*, 24(4):1653–1660, 2018.

[187] Xing-Dong Yang, Tovi Grossman, Daniel Wigdor, and George Fitzmaurice. Magic finger: always-available input through finger instrumentation. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, pages 147–156. ACM, 2012.

[188] Hiroaki Yano, Shoichiro Taniguchi, and Hiroo Iwata. Shape and friction recognition of 3d virtual objects by using 2-dof indirect haptic interface. In *World Haptics Conference (WHC), 2015 IEEE*, pages 202–207. IEEE, 2015.

[189] Vibol Yem and Hiroyuki Kajimoto. Wearable tactile device using mechanical and electrical stimulation for fingertip interaction with virtual world. In *Virtual Reality (VR), 2017 IEEE*, pages 99–104. IEEE, 2017.

[190] Vibol Yem, Mai Shibahara, Katsunari Sato, and Hiroyuki Kajimoto. Expression of 2dof fingertip traction with 1dof lateral skin stretch. In *International AsiaHaptics conference*, pages 21–25. Springer, 2016.

[191] Sang Ho Yoon, Ke Huo, Vinh P. Nguyen, and Karthik Ramani. TIMMi: Finger-worn textile input device with multimodal sensing in mobile interaction. In *Proceedings of the Ninth International Conference on Tangible, Embedded, and Embodied Interaction*, TEI '15, pages 269–272, New York, NY, USA, 2015. ACM.

[192] Sang Ho Yoon, Ke Huo, and Karthik Ramani. Plex: Finger-worn textile sensor for mobile interaction during activities. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, UbiComp '14 Adjunct, pages 191–194, New York, NY, USA, 2014. ACM.

[193] Raimondas Zemblys and Oleg Komogortsev. Making stand-alone ps-og technology tolerant to the equipment shifts. In *Proceedings of the 7th Workshop on Pervasive Eye Tracking and Mobile Eye-Based Interaction*, page 2. ACM, 2018.

[194] André Zenner and Antonio Krüger. Shifty: A weight-shifting dynamic passive haptic proxy to enhance object perception in virtual reality. *IEEE Transactions on Visualization and Computer Graphics*, 23(4):1285–1294, 2017.

[195] Yang Zhang and Chris Harrison. Tomo: Wearable, low-cost electrical impedance tomography for hand gesture recognition. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, pages 167–173. ACM, 2015.

[196] Yang Zhang, Robert Xiao, and Chris Harrison. Advancing hand gesture recognition with high resolution electrical impedance tomography. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pages 843–850. ACM, 2016.

[197] Yang Zhang, Junhan Zhou, Gierad Laput, and Chris 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*, pages 1491–1503. ACM, 2016.