Improving Touch Accuracy for People with Motor Impairments

Martez Edward Mott

A dissertation

submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

University of Washington

2018

Reading Committee:

Jacob O. Wobbrock, Chair

Leah Findlater

David Hendry

Meredith Ringel Morris

Program Authorized to Offer Degree:

The Information School

University of Washington

Abstract

Improving Touch Accuracy for People with Motor Impairments

Martez Mott

Chair of the Supervisory Committee:
Professor Jacob O. Wobbrock
Information School

Touch-enabled devices such as smartphones, tablets, and interactive kiosks are some of the most pervasive technologies in the world today. Despite the overwhelming popularity of touch input, it presents significant accessibility challenges for millions of people with motor impairing conditions such as cerebral palsy and Parkinson's disease, and for people who experience motor difficulties as the result of injury or old age. This dissertation presents an ability-based design approach towards creating touch models that can improve the accuracy of touch input for people with motor impairments. I conducted two exploratory studies to understand the touch behaviors of people with motor impairments when interacting on an interactive tabletop computer and a smartphone. These exploratory studies led to the design of two new ability-based touch models: Smart Touch, a user-specific template matching algorithm designed initially for tabletops; and Cluster Touch, a combined user-specific and user-independent touch model that improves touch accuracy on smartphones. I also conducted an exploration of using convolutional neural networks to model a user's touch behavior to predict intended target locations. This work presents a thorough approach to designing touch models that can support the touch abilities of people with motor impairments.

I will demonstrate the following thesis:

Ability-based touch models can improve touch accuracy on touch screens compared to native touch sensors or existing statistical models for people with motor impairments and for people in motor-impairing situations.

Contents

List of Figures

List of Tables

ACKNOWLEDGEMENTS

I would like to thank my advisor, Jacob O. Wobbrock, for his unwavering support. Jake pushed me to be the best version of myself, always encouraging me to work hard and show pride in my efforts. Jake not only taught me how to be a researcher, he instilled in me the importance and responsibility of being a scholar. Jake has taught me more than he could ever know, and I am forever grateful.

I would like to thank the members of my dissertation committee for their guidance and support. Leah Findlater provided critical insights and expertise. Richard Ladner made me think about the practicality of my methods and inventions. David Hendry encouraged me to step back from the details and consider the bigger picture. Merrie Morris provided fresh ideas and ways to integrate my research with prior work.

I never would have made it to UW without my mentors and friends from Bowling Green State University. BGSU is where I discovered my love for research. I am extremely fortunate that I found a group of professors willing to nurture my curiosities, and fellow students willing to explore with me. Thank you all: Tom Donahue, Darren Guinness, Dale Klopfer, Laura Leventhal, G. Michael Poor, Briana Tomlinson, and Guy Zimmerman. To Duke Hutchings, who first encouraged me to pursue research, I cannot thank you enough; you changed my life.

The University of Washington is a remarkable place, and I am extremely thankful to current and past faculty and staff for making my time at UW so welcoming: Victor Aque, Scott Barker, Miranda Belarde-Lewis, Jason Boyd, Harry Bruce, Mary Clark, Jayson Curry, Negin Dahya, Katie Davis, Cynthia del Rosario, Nick Dempsey, Dowell Eugenio, James Fogarty, Cris Fowler, Alexis Hiniker, Joseph Janes, Julie Kientz, Dean Kirkpatrick, Andy Ko, Cortney Leach, Michelle Martin, David McDonald, Cheryl Metoyer, Adam Moore, Wendie Phillips, Wanda Pratt, Matthew Saxton, Jamie Snyder, Emma Spiro, Kat Steele, Hazel Taylor, Joseph Tennis, Dora Tkach, Jevin West, and Jason Yip.

My friends and fellow colleagues kept me sane during my studies. I could not have done it without them: Norah Abokhodair, Dan Afergan, Elena Agapie, Stephanie Ballard, Cynthia Bennett, Robin Brewer, Patrick Carrington, Mike Depew, Marisa Duarte, Daniel Epstein, Chris Heaney, Nona Ingram, David James, Nekia Hampton, Jeff Hemsely, Mike Katell, Saba Kawas, Ian King, Rose Kinsley, Walter Lasecki, Michael Lee, Lassana Magassa, Elizabeth Mills, Sonali Mishra, Beth Patin,

Caroline Pitt, Jason Portenoy, Kyle Rector, Zithri Saleem, Marc Schmalz, Mina Tari, Katherine Thornton, Beck Tench, Paul Weiss, Benjie Xie, Katya Yefimova, Ivette Bayo Urban, Shawn Walker, Meg Young, and Annuska Zolyomi.

My MAD Lab family has been an invaluable source of advice and support. It is truly a pleasure to be part of such a phenomenal group of people and researchers: Abdullah Ali, Shiri Azenkot, Parmit Chilana, Abi Evans, Rachel Franz, Jeff Huang, Shaun Kane, Alex Mariakakis, Annie Ross, Kristen Shinohara, Jessica Tran, Tony Tung, and Ray Zhong.

The T2 Liquid Metal Cohort is the best cohort I could have ever asked for. Truly the brightest and most kind group of people I have ever met. They made my experience at UW truly special: Hyerim Cho, Zak Dehlawi, Jordan Eschler, Sandy Littletree, Amanda Menking, Philip Reed, and my twin sister, Katie O'Leary.

I would like to thank my iSchool Inclusion Institute (i3) family for their continued support and encouragement. Visiting i3 lifted my spirit and filled my soul, which kept me going during hard times.

To my best friends, Greg Bolden, Elizabeth Harvin, Garrett Jackson, and Jared Mims, thank you for all your love and encouragement throughout the years. You all kept me going.

I would like to thank my parents, Toni and Michael Mott, for instilling in me a strong work ethic, and for always stressing the importance of education. Your continued support made all this possible. Whenever I was feeling down, you both were there to lift me up, dust me off, and remind me that I can do anything. I also want to thank my big brother, Mike, who was always there for me, no matter what, and my big sister, Dodie, for always believing in me.

I would like to thank the staff at Provail for their help and support in conducting my research. A special thank you to Gabriel Laigo for allowing me to set up equipment in his lab, and for helping me recruit participants.

This research was funded in part by the National Science Foundation, Google, Microsoft Research, the University of Washington Information School, the University of Washington Graduate School, and the Mani Charitable Foundation.

DEDICATION

To Nisha

We miss you.

Chapter 1. Introduction

We live in an increasingly digital world, where computing is essential to many of the daily activities people perform. For work, people use computing devices to stay productive and connected while in or away from the office. At home, people play computer games, video chat with relatives, and share photos with friends on social media. Over the past decade, people have done an increasing amount of their computing tasks on mobile computing devices such as smartphones and tablets. According to a 2018 survey from the Pew Research Center, 77% of adults in the United States own a smartphone, while 73% own a laptop or desktop, and 53% own a tablet computer[1].

1.1. Pervasiveness of Touch Input

The popularity of touch-enabled mobile devices (e.g., smartphones and tablets) has resulted in touch emerging as one of the most pervasive input methods for today's computing systems. In addition to personal smartphones and tablets, people often interact with touch screens in public spaces, and across various devices in their home or office. Touch-enabled kiosk are increasingly used in grocery stores to help customers avoid long lines, and at airports to allow travelers to print their own boarding passes. Interactive wall displays provide immersive experiences at museums and tourist attractions—like the SkyPad inside Seattle's Space Needle[2].

Touch screens are also prevalent in various household and office machines. Touch screen panels are common on office copiers and printers, and are becoming more common on refrigerators, microwaves, and other household appliances. Popular wearable devices such as the Apple Watch are also touch-enabled. It is not uncommon to find touch screen devices in ATMs, classrooms, and elevators. During an average day, a person might encounter several unique touch-enabled devices.

1.2. Benefits of Touch Screens

Touch screens can provide benefits over traditional mouse and keyboard input. Unlike a mouse, or other indirect pointing devices, touch is a direct input method. Users can touch the screen to directly interact with on-screen objects, unlike indirect pointing devices, which require users to control an on-screen cursor by manipulating a secondary device (mouse, trackball, etc.). Touch interaction is also

---

1 http://www.pewresearch.org/fact-tank/2018/09/28/internet-social-media-use-and-device-ownership-in-u-s-have-plateaued-after-years-of-growth/

2 http://www.bwco.info/work/skypad/

typically quite fast, and requires less hand-eye coordination and desk space compared to a mouse or keyboard [73]. As a result, touch is typically considered more intuitive than other input methods [9]. The flexibility of touch input makes touch screens a popular choice for many mobile devices and computing devices in public spaces.

1.3. Accessibility of Touch Screens for People with Motor Impairments

Although touch input has been widely adopted, it still poses significant accessibility challenges to many people with motor impairments. Touch screens, like all technologies, are created with implicit ability-assumptions. These ability-assumptions determine how users can interact with systems. For touch screens, there is an implicit requirement that a user must be able to suspend an arm, extend a single finger, and accurately land and lift their finger within the bounds of their intended target. However, for many people with motor-impairing conditions such as muscular dystrophy or multiple sclerosis, these actions may be difficult or impossible to perform. Instead of touching with a single finger, users may touch with multiple fingers, the back of their hand, their elbow, or their nose [2,57]. Instead of landing and lifting their finger precisely on the screen, users may drag their finger across the screen, miss their target because of uncontrollable motions such as tremors, or inadvertently touch with a second finger—accidently triggering actions via multi-touch gestures [28,29,57,74].

Touch screens remain inaccessible to many people with motor impairments because their abilities do not match the ability-assumptions embedded in the design of touch screen devices. This ability mismatch is problematic; as touch-based interfaces continue to grow in popularity, more services and applications are available only on mobile touch-enabled devices. Inaccessible touch interfaces can prohibit users with motor impairments from engaging in preferred behaviors, such as texting or capturing photos on a smartphone [56,59]. Also, as emerging touch-enabled devices such as smartwatches and fitness trackers become mainstream, it is important that people with motor impairments be able to engage with these devices and take full advantage of their capabilities [51].

Previous research has investigated the accessibility challenges people with motor impairments encounter when using touch screens in lab [19,21,22,28,29,37,72,74,89] and field [2,38,55,59] settings. Although this research is helpful to understand difficulties users with motor impairments experience, and how well they complete certain tasks on touch screens, there has been little advancement into new techniques for improving touch performance for these users. Touch Guard by Zhong *et al.* [91] and Swabbing by Wacharamanotham *et al.* [78] are two examples of new touch techniques developed specifically for people with motor impairments. Although these techniques were found to be effective,

they do not consider the different touch behaviors exhibited by people with motor impairments. For example, Swabbing assumes that a user can contact the screen with a single finger, which may not be comfortable or possible for all users [2,57].

1.4. Motivation

As described above, touch screens are often inaccessible because users with motor impairments do not meet the ability-assumptions embedded in touch screens. Prior efforts to improve touch accuracy have focused primarily on creating new touch-based interaction techniques for people with motor impairments [54,78,91]. These techniques have limitations, however, as many of them are designed to accommodate one type of touch input, which is typically a single finger. However, to fully enable touch screens to be accessible by a broad range of users with motor impairments who may use varying touch behaviors to interact with touch screens, a more holistic approach is needed to understand the touch behaviors exhibited by people with motor impairments.

My research takes an ability-based design [82,86,87] approach towards creating ability-based touch models that can improve the accuracy of touch input for people with motor impairments. Ability-based design is a design framework that aims to remove the burden of users having to adapt themselves to match to the rigid ability-assumptions of computer systems. Instead, ability-based design requires systems to adapt to the unique abilities of their users. By flipping the burden of adaption from the user to the system, users can approach systems as they are without first procuring additional technologies to make the system work for them. For touch screens, users with motor impairments may purchase a mouth or head stylus in order to interact with their device. By creating new ability-based touch models that can account for individual abilities, I aim to remove the need for users to procure these secondary objects.

The goal of this dissertation is to improve the accuracy of touch screens for people with motor impairments. In this dissertation, I demonstrate that ability-based touch models can be trained to understand the touch behaviors of users with motor impairments, and, once trained, these models can improve touch accuracy for people with motor impairments, and for people without motor impairments under the effects of situational impairments [69,71]. I devised these touch models by collecting examples of motor-impaired touch, analyzing the touch examples to find similarities and patterns across different users, using the analysis to inform the design of ability-based touch models, and testing the effectiveness of the models using offline testing methods and interactive tests with real users.

For this dissertation, I will demonstrate the following thesis:

Ability-based touch models can improve touch accuracy on touch screens compared to native touch sensors or existing statistical models for people with motor impairments and for people in motor-impairing situations.

1.5. Research Overview

To demonstrate my thesis, I conducted three investigations into building and evaluating ability-based touch models for people with motor impairments. My research occurs in three stages: (1) understanding; (2) development; and (3) evaluation. In the first stage, I often conducted exploratory analysis to better understand how people with motor impairments interact with touch screens. These explorations provide foundational knowledge about the touch behaviors and abilities of people with motor impairments. Understanding these touch behaviors provides insights into how touch systems currently interpret touches performed by people with motor impairments. In the second stage, I use the insights gleaned from my exploratory analyses to build touch models that can improve touch accuracy for people with motor impairments. Building touch models requires an understanding of the underlying touch behaviors of users. Otherwise, the models would not be representative of how people touch, rendering the models ineffective. Constructing these touch models requires multiple rounds of iteration, which include testing different model parameters to see how they impact a model's performance. Finally, in the third stage, my models are evaluated and compared to the native touch sensors found in touch screen devices, and to previously developed touch models. Evaluating these models is important to understand how much touch accuracy can be improved over native touch sensors and other touch models. These evaluations also provide insights into how my models perform for specific users, which is useful feedback when attempting to understand why a model may have performed better for one user and more poorly for another. These insights can then be used to further update and improve the design of the models.

1.6. Structure of this Dissertation

This section describes the remaining structure of this dissertation. Chapter 2 describes related work in the areas of touch screen accessibility and techniques to improve the accuracy of touch screens.

Chapter 3 describes an exploratory study to understand the touch behaviors of people with motor impairments when interacting on an interactive tabletop. Using the data collected from this touch behavior exploration, I created Smart Touch, a novel user-specific template matching algorithm that resolves any number of arbitrary contact points to predict a user's intended (x, y) touch location.

Chapter 4 describes an exploratory study to understand the touch offsets created by people with motor impairments, and people without motor impairments while walking, when interacting with a smartphone touch screen. From this analysis I developed Cluster Touch, a combined user-independent and user-specific touch model that improves touch input with only 20 touch examples provided by the user.

Chapter 5 describes an exploration into using convolutional neural networks [46] to build models of users' touch behaviors in order to improve touch accuracy. This chapter provides an analysis into how well a trained convolutional neural network could predict intended target locations. This chapter also discusses implementation details to consider when training convolutional neural networks for touch modeling, and what is needed moving forward to fully explore this topic.

Chapter 6 provides a discussion on the practical implications of using touch models to improve touch accuracy. I discuss the benefits and limitations of these approaches, as well as the privacy implications that arise when creating and deploying touch models in interactive systems.

Chapter 7 highlights the contributions of this dissertation and discusses avenues for future work.

Chapter 2. Related Work

This chapter discusses related work in the areas of accessible touch screen interaction for people with motor impairments, high precision touch techniques, probabilistic and statistical touch input, theories of touch input, and ability-based design.

2.1. Touch Screen Accessibility for People with Motor Impairments

Researchers have investigated numerous aspects of touch screen accessibility for people with motor impairments. One area of research is to understand how people with motor impairments use and interact with touch screens in their daily lives [2,38,59]. In these investigations, the authors found that mobile touch screen devices can provide a sense of freedom and empowerment to users with motor impairments [38,59]. Anthony *et al.* [2] analyzed YouTube videos featuring people with motor impairments interacting with touch screens. The authors found that users with motor impairments adopted various kinds of interaction styles, including touching with multiple fingers, the backs of their hands, fists, and knuckles. Naftali and Findlater [59] found that smartphones provided multiple benefits to people with motor impairments, including using a smartphone to mitigate the accessibility challenges they encounter in the world. The authors also found that their participants experienced difficult when performing common smartphone interactions, such as tapping and entering text with a keyboard.

Other researchers have investigated touch screen use in the lab. Notably, Guerreiro *et al.* [28,29] explored how well people with motor impairments could perform interactions such as tapping and crossing on a mobile phone. They found that users experience difficulties interacting with certain screen regions, and that selecting small targets was difficult or impossible for some users. Trewin *et al.* [74] found that performing sliding and tapping gestures resulted in inaccurate touch locations and the accidental activation of other touch screen gestures, such as zooming. They also found that some participants would use the screen for stability, sliding their fingers on the screen towards their intended targets. In their investigation of mouse and touch input, Findlater *et al.* [21] found that users with motor impairments were significantly less accurate using a touch screen compared to a mouse. In their exploration of motor-impaired touches, Montague *et al.* [55] found that the tap gestures performed by people with motor impairments were often misclassified as other touch-related gestures, or were disregarded all together.

Much of the current research in accessible touch input for people with motor impairments has been descriptive (highlighting the accessibility challenges experienced by people with motor impairments in natural and controlled settings) and prescriptive (providing suggestions to improve accessibility, such as increasing the size of touch widgets). However, little work has been done in creating new interaction techniques or models of touch behavior to improve touch accuracy for people with motor impairments. Biswas and Langdon [11] created an algorithm that measures hand strength to improve pointing across several input methods, including touch. Montague *et al.* [55] developed a novel gesture recognizer to improve tap recognition on touch screens for users with motor impairments. Montague *et al.* [54] also created the Shared User Modeling Framework, which is a framework aimed at improving touch interaction for people with motor impairments across different devices and applications. Swabbing is an interaction technique for older adults developed by Wacharamanotham *et al.* [78]. With Swabbing, users drag their fingers across their intended targets instead of performing discrete tap gestures. Zhong *et al.* [91] created Touch Guard, a system that disambiguates targets by magnifying targets near where the touch occurred, or by showing possible targets in list form.

My dissertation expands on previous investigations into the accessibility of touch screens for people with motor impairments by providing ability-based touch models. Instead of specific interaction techniques that require users to conform to a particular touch style (e.g., touching with a single finger), my touch models provide options to allow people to touch in whichever way is most comfortable and natural for them. My touch models also support users without motor impairments while under typical and situationally-induced impairments [69,71].

2.2. High Precision Touch Techniques

Accurately selecting targets on a touch screen has been a problem since touch screens' inception, thanks in large part to the "fat finger problem" [76,77]. Unlike a mouse cursor, whose hot spot occupies a single pixel, a finger occupies an entire area. Thus, the system must infer exactly where in this area the user is intending to touch. Various interaction techniques have been devised to overcome this limitation. Potter *et al.* [63] investigated the effectiveness of the offset-cursor, an on-screen cursor that appears above the user's finger when they contact the screen. Sears and Shneiderman [70] improved the performance of the offset-cursor by adding stabilization. Cross-Keys and Precision Handle are two interaction techniques created by Albinsson and Zhai [1] that allow for pixel-level precision pointing. Dual Finger Selections is a collection of five interaction techniques proposed by

Benko *et al.* [5] that use bimanual interaction to acquire small targets. Shift by Vogel and Baudisch [77] is an interaction technique that identifies when a finger occludes a small target and creates a callout to display the occluded target on a non-occluded region of the screen, which allows small targets to be acquired more quickly and accurately. LucidTouch by Wigdor *et al.* [81] circumvents the fat finger problem by allowing users to select targets by touching the back of the screen.

Although these techniques have been shown to improve fine pointing performance, none of these techniques have been tested with users with motor impairments. Some of these techniques, such as the Dual Finger Selections, require a great deal of dexterity, which a user with a motor impairment might not possess. While individual interaction techniques may work well for some users, it can be difficult to create interaction techniques that will work for a user population that exhibits a great deal of variation in touch behaviors from person to person. For example, participants with adequate manual dexterity may be able to use one of the Dual Finger Selection techniques, but these techniques would be improbable for users who touch with multiple fingers or various parts of their hand. Thus, there is a benefit to creating touch models that can accommodate a wide range of possible touch behaviors. My dissertation demonstrates that ability-based touch models can afford users with motor impairments greater pointing accuracy while allowing them to touch the screen however their abilities allow.

2.3. Probabilistic and Statistical Touch Input

Touch is an uncertain form of input. Because a touch represents an entire area, where a user is intending to touch is left to the system to interpret. Researchers have attempted to leverage the uncertain nature of touch input to improve touch accuracy. Schwarz *et al.* created an architecture [68], a framework [66], and new methods [67] for handling uncertain inputs such as touch. In their investigations, Schwarz *et al.* [66–68] found that touch interaction could be improved by considering factors other than the reported touch locations. For example, user interfaces could consider the probabilities of performing different actions when determining how to interpret touch input. A touch that "hit" a low probability target could be corrected if the system infers that the touch was intended for a target with a higher probability of selection.

Bayesian Touch by Bi and Zhai [9] is a statistical target selection technique that leverages the bivariate Gaussian distribution principle of finger touch [8,10] to predict users' intended targets. In their investigations, Bi and Zhai found that touch follows a bivariate Gaussian distribution, where one

univariate distribution is the result of the absolute accuracy of the pointing device (i.e., a finger), and the other univariate distribution is the relative accuracy of the user's touch behavior. ProbUI by Buschek and Alt [13] replaces static target bounding regions with probabilistic gestures, allowing users to perform a wide array of actions on otherwise static targets.

Weir *et al.* [79,80] treat touch as a machine learning problem, and used Gaussian process regression to build user-specific models of touch. They found that user-specific touch models outperformed touch models trained with data from other users. Buschek *et al.* [14] also treated touch as a machine learning problem, and investigated the effectiveness of a linear offset model to improve touch accuracy across different devices and across different users. Similar to Weir *et al.* [80], they found that user-specific models outperformed general models. The TouchML toolkit by Buschek and Alt provides access to these different models, as well as touch examples collected from prior studies [12].

Probabilistic input methods have the potential to improve touch accuracy, but they also require systems to be target-aware [4]. Target-aware systems know the location, dimensions, and states of all targets shown on the screen. These systems are difficult to implement in practice, as what constitutes a "target" is not always well-defined [18]. The ability-based models in this dissertation are target-agnostic [85], meaning they do not require any prior knowledge about on-screen targets. As UI frameworks continue to evolve, probabilistic techniques may have the potential to increase touch performance, but further research is needed to see how the touch behaviors of people with motor impairments interfere with the ability-assumptions currently embedded in probabilistic techniques.

Treating touch as a machine learning problem has yielded touch models able to improve touch accuracy over native sensors. However, touch behaviors produced by some users with motor impairments are not suitable for these machine learning models. For example, touching with multiple fingers or various parts of the hand will create multiple concurrent touches, but these models—as currently constructed—cannot accommodate that type of touch behavior. This dissertation introduces ability-based touch models, which are target-agnostic, and can accommodate a variety of different touch behaviors.

2.4. Touch Input and Situational Impairments

Situationally-induced impairments and disabilities, or situational impairments for short, are caused by environmental factors that temporarily affect users' ability to interact with computing devices [69,71]. Everyday activities such as walking or carrying a bag could have adverse effects on touch accuracy

[6,16,60]. Bergstrom *et al.* [6] found that touch accuracy always declines while users are walking, and that the errors increase in severity the faster uses walk. Ng *et al.* [61] found that participants were less accurate performing tapping and dragging gestures when encumbered. Lin *et al.* [47] found that tapping accuracy was significantly reduced while walking. Kane *et al.* proposed "walking user interfaces" [40] to enlarge targets while users walk to maintain touch accuracy. WalkType by Goel *et al.* [27] is a technique to improve text entry on smartphones by leveraging accelerometer data reported by the device. Musić *et al.* [58] showed that users' gait phase can be used as an additional input to an offset model to improve touch accuracy.

These investigations show that various situational impairments negatively impact touch accuracy. These investigations also show that techniques can be developed to improve touch accuracy for users while under different situational impairments. However, it is unknown whether or not techniques proposed to improve touch accuracy for users under situational impairments would also benefit users with motor impairments. My dissertation adds to this body of knowledge by creating and evaluating Cluster Touch, a smartphone-based touch model that improves accuracy for users under the effects of situational impairments, and for users with motor impairments. My work demonstrates that models created for one user group may also provide some benefits to users with varying abilities and in varying contexts.

2.5. Theories of Touch Input

Holz and Baudisch [35,36] conducted two studies to understand users' perception of touch. In their first study [35], they introduced the "generalized perceived input point model," which states that the primary source of touch input error is not caused by the fat finger problem, but is caused by users' perception of touch input and the systematic errors they commit when producing touches. The authors found that applying a reverse touch offset could be used to correct touch inputs. In their second study [36], the authors explored how users perceive touch input by asking them to perform touches using different models based on visual finger properties. They found that users tended to align touches with certain visual properties of their finger, primarily the center of their finger tip.

These investigations provide evidence that users' mental models of touch affect their touch behaviors and ultimately impact touch accuracy. However, since these investigations were conducted with nondisabled users, we cannot infer how well the proposed models would work for people with motor impairments, or if the mental models of touch possessed by the participants in their two studies

differed from the mental models of touch possessed by people without motor impairments. My investigations into building ability-based models provides an understanding of the kinds of touch behaviors expressed by people with motor impairments. Although my studies cannot speak to how users themselves perceive their touch behaviors, collecting data from touch screens provides useful information into understanding how users express their touches.

2.6. Ability-Based Design

My dissertation is motivated by the concepts and principles of ability-based design [82,86,87]. In ability-based design, the emphasis is placed on discovering what users can do, instead of focusing on what users cannot do. A key aspect of ability-based design is that the burden of adaptation should be placed on the system, rather than the user. One of the best examples of ability-based design is the SUPPLE system by Gajos et al. [23–25]. SUPPLE automatically generates user interfaces to accommodate the unique pointing behaviors of users. While other design principles such as universal design [50] and inclusive design [41,42] also motivate designers and researchers to create accessible applications and devices, ability-based design demands a tighter fit between the abilities of the user and the underlying interface. By tailoring interfaces and techniques not to a broad group of users, but to one specific user, we can create interventions that greatly improve the experience for that user.

In my dissertation, I combine knowledge of users' touch abilities (the ability and adaptability principles) to create ability-based models able to improve touch accuracy for people with motor impairments (the accountability and performance principles).

Chapter 3. Smart Touch: Improving Touch Accuracy with Template Matching*

This chapter provides details on an exploratory study conducted to understand the touch behaviors exhibited by people with motor impairments when touching an interactive table. This chapter also introduces Smart Touch [57], a user-specific touch model that resolves users' intended touch locations while allowing people with motor impairments to touch in whichever way is most comfortable and natural for them.

3.1. Motivation

In their investigation of YouTube videos featuring people with motor impairments interacting with touch screen devices, Anthony *et al.* [2] found that people with motor impairments used a variety of different body parts to interact with the screen. Little was known, however, about the touch behaviors of people with motor impairments who touch with multiple fingers or various parts of their hand. Performing touch behaviors described by Anthony *et al.* [2] results in inaccessible touch interactions.

Touch screens presume users can perform actions such as suspending an arm, extending a single finger, and landing and lifting accurately within the bounds of their intended target. However, prior works have found that not all people with motor impairments can perform these actions [2,74]. My perspective is that the inaccessibility of touch screens does not inherently lie with users, but with the implicit ability-assumptions embedded in the design of touch screen devices. It is due to these ability-assumptions that do not match the abilities of all users that cause touch screens to be inaccessible to people with motor impairments. I adopted an ability-based design [82,86,87] approach towards investigating how people with motor impairments interact with touch screens.

The contributions of this work are two-fold. First, I conducted an exploratory analysis of the touch behaviors exhibited by ten people with motor impairments. The purpose of this investigation was to uncover how the touch behaviors of users with motor impairments impacted their ability to acquire targets on a touch screen. Second, I created and evaluated Smart Touch, a novel three-step user-specific touch model that maps any number of arbitrary contact areas to a user's intended (x, y)

* Parts of this chapter are adapted from: Mott, M.E., Vatavu, R.-D., Kane, S.K., and Wobbrock, J.O. (2016). Smart Touch: Improving touch accuracy for people with motor Impairments with template matching. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '16), 1934–1946.

touch location. My evaluation of Smart Touch found that with 30 touch examples, Smart Touch was significantly more accurate in predicting touch locations compared to the default sensors found in the Microsoft PixelSense interactive table.

3.2. Exploration of Motor-Impaired Touch

To further my understanding of how people with motor impairments interact with touch screens, I conducted a preliminary study where participants were asked to touch the center of crosshairs displayed on a Microsoft PixelSense interactive tabletop.

					Self-reported impairments†										
ID	**Age**	**Sex**	**Touch method**	**Health condition**	**Mo**	**Sp**	**St**	**Tr**	**Co**	**Fa**	**Gr**	**Ho**	**Se**	**Dir**	**Dis**
1	61	M	Fist	Cerebral Palsy	✓	✓	✓		✓		✓			✓	✓
2	37	F	Fingers	Cerebral Palsy	✓	✓	✓			✓	✓	✓		✓	✓
3	42	F	Fingers	Spinal Cord Injury	✓		✓		✓	✓	✓	✓	✓	✓	✓
4	47	M	Fingers	Cerebral Palsy	✓		✓		✓	✓	✓	✓		✓	✓
5	58	M	Fingers	Cerebral Palsy	✓		✓	✓	✓	✓	✓	✓		✓	✓
6	55	M	Hand	Cerebral Palsy			✓		✓	✓	✓	✓		✓	✓
7	63	F	Fingers	Cerebral Palsy	✓	✓	✓		✓	✓	✓	✓		✓	✓
8	51	F	Fingers	Cerebral Palsy	✓		✓	✓	✓	✓	✓	✓		✓	✓
9	59	M	Fingers	Multiple Sclerosis		✓	✓		✓	✓	✓	✓	✓		
10	52	M	Fingers & Hand	Cerebral Palsy	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

† Mo = slow movements, Sp = spasm, St = low strength, Tr = tremor, Co = poor coordination, Fa = rapid fatigue, Gr = difficulty gripping, Ho = difficulty holding, Se = lack of sensation, Dir = difficulty controlling direction, Dis = difficulty controlling distance.

Table 1. Demographic information for my participants. Categories used for self-reporting impairments were from Findlater et al. [20]

3.2.1. Participants

I recruited 10 people (4 female, 6 male, average age 52.5 years, *SD*=8.46) with motor impairments to participate in my study. Participants were recruited from Provail[3], a Seattle-based organization that

[3] https://provail.org/

provides rehabilitation, job placement, and community living for people with physical disabilities. Four participants reported having more motor control in their left arm and hand, and six participants reported having more motor control in their right arm and hand. Participants were paid $30 for participating in the study session, which lasted approximately one hour. Additional participant details can be found in Table 1.

3.2.2. Apparatus

Experiment sessions were conducted on a Microsoft PixelSense interactive table running Windows 7. I selected the PixelSense to conduct this study because the PixelSense provides touch data not commonly found on most platforms. A custom experiment testbed was developed in C# .NET 4.5 that collected all touch data produced during the study. Each touch registered by the PixelSense was presented as an ellipse with a major and minor axis, an orientation, and a center represented by an x- and y-coordinate. The PixelSense was placed on an adjustable height table to accommodate participants in wheelchairs. The table was adjusted in each session to a height comfortable for each participant.

Figure 1. A participant touches a crosshairs on the Microsoft PixelSense.

3.2.3. Procedure

Seven participants completed the study in an accessible computer lab at Provail, and three participants completed the study at the University of Washington. Each session began with the experimenter describing the capabilities of the PixelSense and the information that would be collected during the study. Next, participants were asked to demonstrate where on the screen they were most comfortable touching. Using a mouse, the experimenter then drew a rectangle bounding the screen area where the

participant indicated they were comfortable touching. The rectangle served as the interactive region, as crosshairs only appeared inside this region. The location and size of the interactive region varied by participant.

A crosshairs was then drawn inside the interactive region. Participants were instructed to touch the center of the crosshairs in whichever way was most comfortable and natural for them. Each participant was asked to demonstrate touching the crosshairs five times for practice. After the practice touches, data collection began. For each trial, a crosshairs was randomly placed inside the interactive region. A trial started when the first touch event was registered and ended when all registered touches had been removed and no new touches were introduced for one second. The one second timeout was implemented to prevent trials from advancing prematurely, as the touch behaviors exhibited by my participants often resulted in touches that would disappear but reappear shortly after. A three second countdown was provided between trials to allow participants to rest between trials. Participants were asked to complete trials at a comfortable pace. Based on pilot testing, the maximum number of trials was set to 110. Many participants, however, could not complete all 110 trails due to fatigue. Participants completed 94.4 trials on average.

3.3. Results of Touch Exploration

ID	Trials	TD (cm)	TU (cm)	CT	Time (ms)
1	100	27.03 (8.68)	34.51 (7.09)	2.06 (0.65)	3528.16 (2123.80)
2	110	7.29 (3.72)	7.17 (4.13)	1.90 (0.66)	259.57 (302.61)
3	110	8.89 (3.26)	8.09 (3.16)	1.89 (0.48)	398.49 (518.51)
4	82	1.64 (0.83)	1.98 (0.88)	1.00 (0.00)	1841.21 (1925.17)
5	76	7.93 (2.17)	8.16 (2.50)	2.93 (0.98)	2520.36 (1064.16)
6	50	17.14 (3.56)	14.40 (4.96)	3.10 (0.73)	8477.38 (5679.49)
7	84	1.43 (1.10)	0.76 (1.10)	1.35 (0.38)	867.46 (175.40)
8	110	5.50 (4.89)	5.00 (5.01)	1.66 (0.45)	235.15 (210.88)
9	110	7.56 (5.23)	6.84 (5.59)	1.92 (0.47)	290.53 (191.90)
10	100	14.13 (6.34)	12.94 (4.62)	5.48 (1.01)	4206.29 (4343.25)

Table 2. The number of trials completed by each participant, the mean touch-down (TD) and mean touch-up (TU) distance to target center, the mean of the average number of concurrent touches per trial (CT), and the mean trial duration (Time). Standard deviations are shown in parentheses.

I collected a total of 944 trials from my ten participants. Eleven trials were discarded due to a sensor error in reporting participants' lift-off locations. A total of 932 trials were used for analysis (see Table

2 for a breakdown of number of trials completed per participant). The following sections detail the results of my analysis of touch behaviors exhibited by people with motor impairments.

3.3.1.Concurrent Touches

Concurrent touches are touches that occur simultaneously with other touches. To determine the number of concurrent touches, each trial is partitioned into discrete frames (frames will be explained in detail in Section 3.4), and the number of touches in each frame is counted. The average number of concurrent touches was 2.30 (*SD*=1.38). Many participants averaged more than one concurrent touch per trial. Concurrent touches were prevalent because participants used multiple fingers (see Figure 4a) and various parts of the hand (see Figure 4b). Figure 2 shows a distribution of the average number of concurrent touches per trial for each participant.

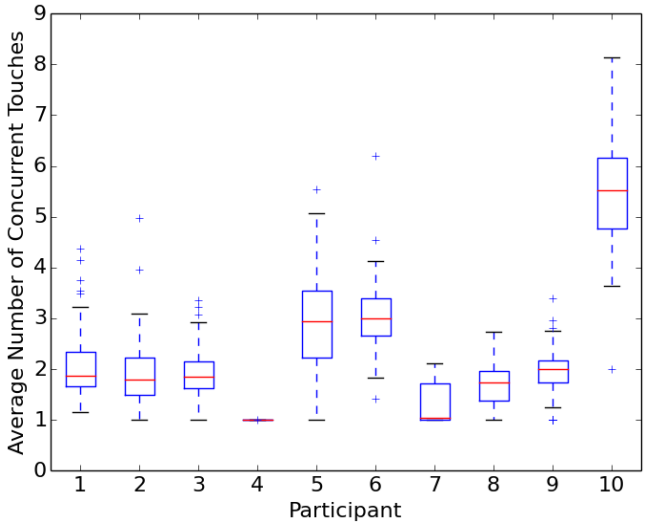

Figure 2. Distribution of the average number of concurrent touches per trial. Due to their touch behaviors, participants impacted the screen with various parts of the hand, resulting in multiple registered touches.

3.3.2. Trial Duration

Trials began when the screen registered a touch and ended when all touches were removed for at least one second. The average trial duration for my participants was 1870 ms (SD=3029). These results show that touches for people with motor impairments are much longer than what is reported in the literature for non-disabled users (600 ms to 1200 ms depending on target size [62]). Mean trial durations for all participants can be found in Table 2.

3.3.3. Touch-Down and Touch-Up

Current touch systems require users to both land and lift within the bounds of their intended targets. To understand how accurately people with motor impairments can land and lift near their intended targets, I measured the distance from the center of the crosshairs to the centers of the ellipses that represent the first and last registered touches for each trial. The mean touch-down distance for my participants was 9.71 cm (SD=8.69), and the mean touch-up distance was 9.97 cm (SD=10.25). Mean touch-down and touch-up distances can be found in Table 2, and Figure 3 shows the distribution of touch-down and touch-up distances for each participant.

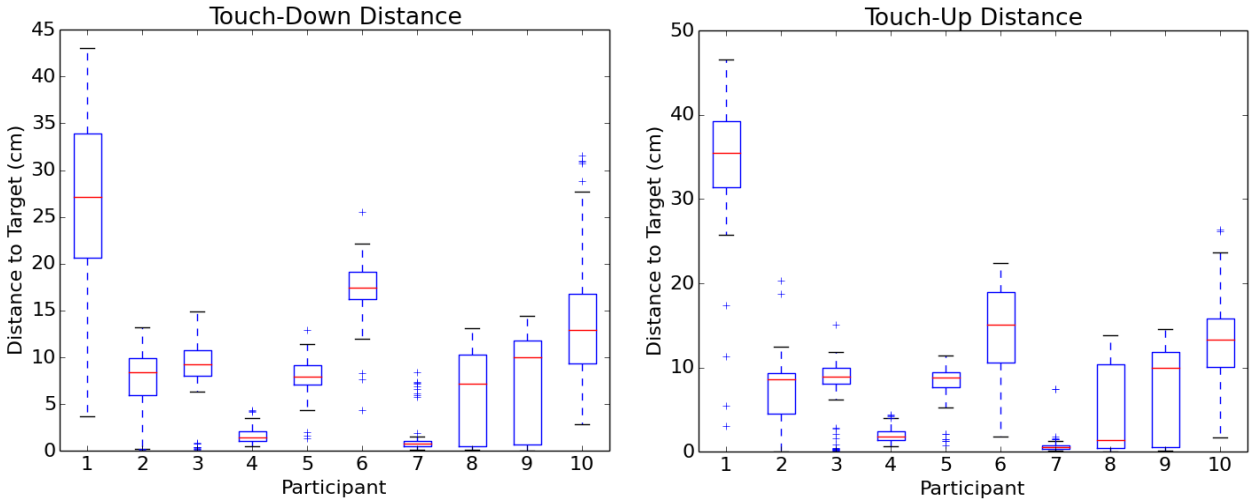

Figure 3. Distribution of distances between the touch-down (left) and touch-up (right) locations to the target center. Note that the distances are measured in centimeters. A centimeter on the Microsoft PixelSense is equivalent to about 22 pixels.

3.3.4. Discussion

The results of my exploratory investigation of touch behaviors of people with motor impairments yielded some interesting results. I found that my participants' touch-down and touch-up locations were significantly further away from the center of the crosshairs when compared to previous studies with non-disabled participants. For example, the average error distance for participants without impairments was 4 mm in a study conducted by Holz and Baudisch [36]. The average touch-down and touch-up distances for my participants were approximately 97 mm and 99 cm, respectively.

The higher touch distances can be attributed to two behaviors observed during the study. The first observed behavior is that participants prematurely touched the screen with their palm or fingers before they were near the crosshairs. Participants also accidently impacted the screen as they lifted their hands or fingers from the screen, resulting in touch-up locations far from the crosshairs.

Accidently impacting the screen is a behavior also observed in prior investigations of smartphone accessibility for people with motor impairments [74]. The second observed behavior is that participants would use the screen to stabilize their hands or fingers while touching. Participants would use the screen for support as they slid their hands or fingers toward their targets. They would also use the screen for support as they moved their hands back towards their bodies before lifting off the screen. Participants who used the screen for support had difficulty moving their arms and hands freely in the air. These two behaviors resulted in touch-down and touch up locations that occurred far from intended targets. These touch distances were much larger than the width of current touch screen widgets. For example, an average button on the Microsoft PixelSense has a width of 2.31 cm, only 23.79% of the average touch-down and touch-up distances for my participants. Even prior recommendations on target sizes for people with motor impairments would not have been enough to accommodate the touch behaviors exhibited in the study. For example, prior work from Guerreiro *et al.* [29] suggested a minimum target size of at least 12 mm.

Another behavior I observed was that participants would use multiple fingers and various parts of their hand to interact with the screen. Some participants attempted to use a single finger but lacked the dexterity to do so. Instead, when these participants used multiple fingers (Figure 4 left) and various parts of their hands (Figure 4 right), the screen would report multiple concurrent touches (Figure 5). The system attempts to fit contact points to the various parts of the hand that touches the screen, resulting in registered contact areas that vary in shape and location (Figure 5).

Figure 4. Participants exhibited multiple touch behaviors, including touching with multiple fingers (left) and with a closed hand (right).

Current touch systems can not currently accommodate varied multi-touch input for simple target acquisition tasks. Prior work has found that when users interact with touch screens using multiple fingers or various parts of their hand, multi-touch gestures are accidentally triggered [74].

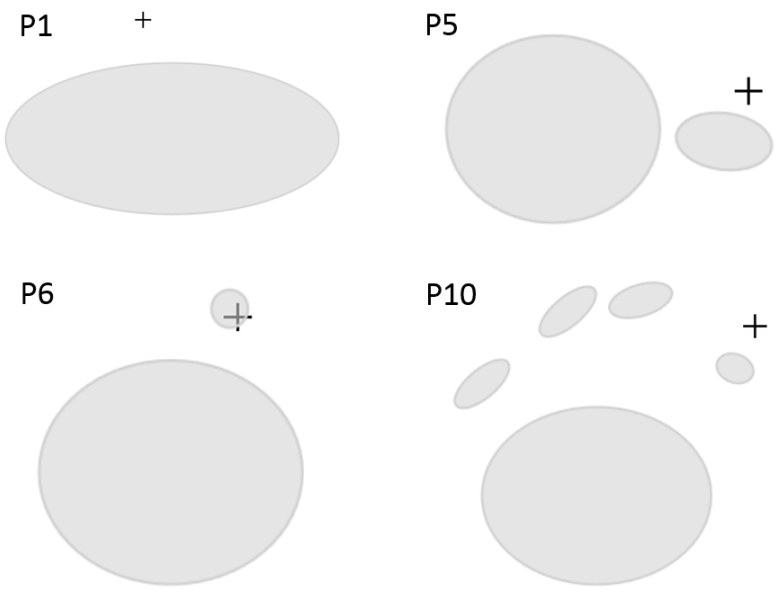

Figure 5. Touch input frames from participants P1, P5, P6, and P10. Ellipses represent contact areas captured by the PixelSense.

To improve touch screen accuracy for people with motor impairments, it is important to view the touch behaviors of people holistically. Current touch screen systems place a heavy emphasis on the first and last contact points a user makes with the screen, as the system presumes that users can accurately land and lift within the bounds of their intended target. Current systems also presume that users will touch with a single, fully extended finger, which would result in the system only registering one unique touch point. However, the results of my exploratory study show that touching with a single, fully extended finger and accurately landing and lifting near intended targets is difficult for many people with motor impairments. Instead of focusing on the touch-down and touch-up locations, systems need to consider a user's entire touch process. The touch process includes *all* the touch events that occur from the initial touch-down to the final touch-up.

From this investigation, I took away design principles that led me to the creation of Smart Touch, a three-step user-specific template matching algorithm that takes advantage of a user's touch process to improve touch accuracy for people with motor impairments.

3.4. Design of Smart Touch

Smart Touch is a three-step user-specific template matching algorithm that performs three actions. First, Smart Touch analyzes a user's touch process to extract the most relevant touch data. Second, Smart Touch matches the extracted data to previously observed training examples. Third, Smart

Touch predicts the user's intended (x, y) touch location. Smart Touch uses template matching of touch patterns to match the extracted touch data with the previously observed touch instances. Intended touch locations are predicted by adding an (x, y) offset vector associated with the best-matched template to the weighted-centroid of the extracted touch data. This section describes the design of Smart Touch in detail.

3.4.1. Pose Extraction

The first step in Smart Touch is to analyze the users' touch process to find and extract the most relevant touch data. The touch process starts when a user touches the screen and ends when all registered touches have been removed. The touch process for people with motor impairments tends to be longer than the touch process for non-disabled users, as the touch duration for people with motor impairments is typically longer. The touch process for people with motor impairments also tends to have multiple concurrent touches. My analysis of touch behaviors showed that systems cannot rely solely on the touch-down and touch-up locations to infer users' intended touch points. Instead, we must rely on all of the information provided throughout a user's touch process. The steps for pose extraction are explained below.

3.4.1.1. Step 1: Deconstruct the Touch Process into Frames

First, the touch process must be deconstructed into frames (Figure 6). A frame contains all of the touch data (i.e., all of the properties of the ellipses that represent touches registered by the PixelSense) present in a given moment throughout the touch process. New frames are created every time a new touch is added, lost, or modified. A modification occurs when the properties of a registered touch change, such as the center of the ellipse moves.

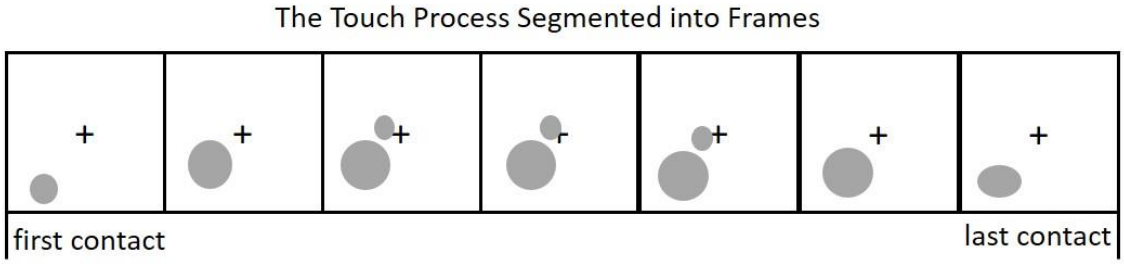

Figure 6. Each frame represents a unique point in time in the user's touch process. Every time a touch is added, changed, or lost, a new frame is created. This figure displays a typical touch process performed by our participants.

3.4.1.2. Step 2: Evaluate the Stability of Touch Frames

After the user's touch process has been partitioned into frames, Smart Touch must decide which frames to extract the touch data to use for matching. From the exploratory study, I noticed that participants tended to dwell when they approached their targets. Thus, I decided that the best time to extract touch data was during this short dwelling period. Dwelling causes touches in those frames to share similar touch properties with one another. Frames that exhibit small differences from the frames that precede them are called stable. Conversely, frames that vary greatly from the frames that precede them are unstable. Frames have two stability properties: movement and shape.

Two stability scores, one for movement and one for shape, are assigned to each frame. Movement stability is based on the difference between the locations of the weighted-centroids of the ellipses that represent the touches in the frame being scored and the frame that precedes it. Shape stability is based on the sum of the area of the ellipses that represent touches in the frame being scored and the preceding frame. A frame is considered movement- or shape-stable if the movement and shape scores are less than 3% of the sum of all movement and shape scores for all the frames in the touch process. I found that a 3% cutoff worked well in testing. A frame that is both movement- and shape-stable is labeled as stable (Figure 7).

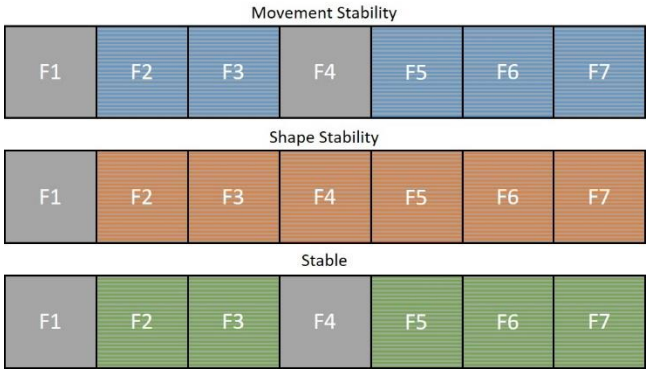

Figure 7. Each of seven frames is labeled stable (highlighted) or unstable (gray) with respect to movement and shape. Frames that are both movement- and shape-stable are categorized as stable overall.

3.4.1.3. Frame Selection

Once all frames have been categorized as stable or unstable, Smart Touch iterates over all the frames to find consecutive stable frames. The lifespan of each consecutive frame is calculated by subtracting the timestamps of the last and first frames in each set. Touch data is extracted from the frame that occurs midway (with respect to time) through the set of stable frames with the longest lifespan (Figure 8). The touch data extracted from the selected frame is called the "indicative pose." The touch data in

the indicative pose is sent to a template matcher to compare the indicative pose to previously observed touch examples. The next section details the pose-matching algorithm.

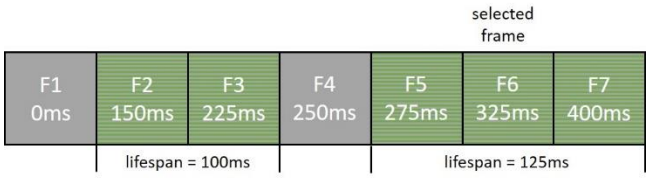

Figure 8. The lifespan of each set of consecutive stable frames is calculated. Touch data is extracted from the frame that is present halfway through the set of stable frames with the longest lifespan.

3.4.2. Pose Matching

After extracting the indicative pose from the user's touch process it is time to compare it to templates of previously extracted poses (a pose is a frame of extracted touch data). This section details the steps in the pose matching process.

3.4.2.1. Translating the Pose

All extracted touch data—both the touch data serving as the template and the touch data serving as the candidate—are translated to a common reference point. First, a bounding box is fit over all touch data for the candidate and the template. Next, the top-left corner of the bounding box is translated to the origin (0, 0). The purpose of translation is to allow touches to be compared regardless of where they occurred on the screen (Figure 9).

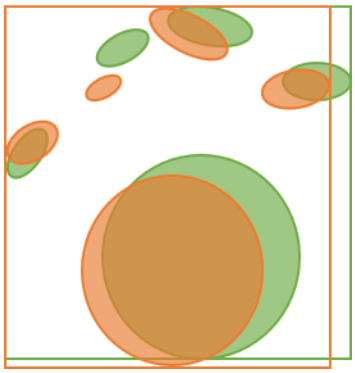

Figure 9. Touch data from two poses translated to the origin, prepared to undergo the template-matching process.

3.4.2.2. Extending $P from Point-Matching to Ellipse-Matching

After the candidate pose *C* and the templates *T* have been translated to the origin, the next step is to find the template that most closely resembles the candidate. To facilitate this matching process, I extended the $P point-cloud stroke gesture recognizer [75]. $P decomposes stroke gestures into point-

clouds and uses a nearest-neighbor point correspondence algorithm to match a candidate gesture to a set of template gestures. To match the candidate point-cloud *C* to a template point-cloud *T*, a function matches each point in *C* to exactly one point in *T*. The goodness of the function is the sum of Euclidean distances for all pairs of points (Eq. 1).

$$\sum_{i=1}^{n} ||C_i - T_j|| = \sum_{i=1}^{n} \sqrt{(C_i.x - T_j.x)^2 + (C_i.y - T_j.y)^2} \quad (1)$$

My template matcher uses the basic underlying behaviors of $P to match each touch instance found in *C* to a touch instance in *T*. Before $P could be used for ellipse-matching, two changes to $P were necessary. First, the template matcher had to accommodate the shape properties of ellipses. Unlike a point, which consists only of an (*x, y*) coordinate, an ellipse has a major and minor axis, an orientation, and a center (*x, y*) location. Thus, the scoring function cannot simply ignore the properties of touches when determining the distance between two poses. To account for the ellipse properties, I extended $P's Euclidean distance formula. The new formula defined distance as the amount of work required to transform one ellipse into another. Therefore, the distance *D* between two ellipses *a* and *b*, is the sum of the Euclidean distance between their centers, the difference between their major and minor axes, and the angular distance in degrees [85] between the orientation of the two ellipses (Eq. 2).

$$D = w_1 * \partial + w_2 * M + w_3 * m + w_4 * \theta, \quad (2)$$

where $\sum w_i = 1.00$, and $\partial = \sqrt{\left[(a_x - b_x)^2 + \left(a_y - b_y\right)^2\right]}$, where *x* and *y* are the center x- and y-coordinates of the ellipse, $M = |a_M - b_M|$, where *M* is the major axis length, $m = |a_m - b_m|$, where *m* is the minor axis length, and $\theta = |(|180 - a_\theta + b_\theta| \bmod 360) - 180|$, where θ is the ellipse orientation in degrees. Optimizing weights resulted in roughly equal weighting, so for simplicity, each w_i was set to 0.25.

The second $P extension was to allow it to accommodate instances when the candidate pose and the template have an unequal number of ellipses. $P originally resampled stroke gestures so that the candidate and the template gesture always had the same number of points, which allowed each point in the candidate to match exactly one point in the template. Resampling touches would not work in this case, as removing or adding touches would result in a loss of valuable information about the user's touch process. Also, unlike stroke gestures, my participants' touches do not have a path through

space to consider. Because it could not be guaranteed that *C* and *T* would have the same number of touches, I created a recursive version of $P that allows for an unequal number of ellipses between the candidate and the template. The algorithm allows ellipses in one pose to serve as the best-matched template for multiple ellipses in the other pose. Poses with different numbers of touches are penalized, and ellipses that go unmatched also increase the distance score for the comparison.

An *N*-best list is generated at the end of the pose-matching process. The template with the lowest score represented the best-match to the candidate. After the best-matched template has been found, the candidate is translated back to its original position on the interactive touch table. Next, Smart Touch predicts the location of the user's intended touch point.

3.4.2.3. Predicting the User's Intended Touch Point

An (*x, y*) vector offset from the center of the weighted-centroid of the pose to the center of the intended target is stored in each template (training examples presented crosshairs to users, so the intended target location is known during training). To predict users' intended touch locations, the (*x, y*) offset vector embedded in the best-matched template is added to the weighted-centroid of the candidate. The resulting (*x, y*) location is Smart Touch's predicted target location.

3.5. Evaluation of Smart Touch

To determine how well Smart Touch could predict users' intended targets, I performed an evaluation of Smart Touch using the data collected from the ten participants from my exploratory study. Using Smart Touch, one pose per trial was extracted for each trial completed by the ten participants. Each participant had the same number of extracted poses available for testing as the number of trials completed by the participant in the exploratory study.

3.5.1. Design and Analysis

I first conducted an initial experiment to determine how many templates were needed in order for Smart Touch to improve touch accuracy. The initial experiment was a within-subjects design with the following factors and levels:

- Number of training examples *E*: 10, 20, 30, 40

The second experiment compared the accuracy of Smart Touch to the land-on and lift-off distances reported by the touch sensors in the Microsoft PixelSense. The second experiment was a within-subjects design with following factors and levels:

- *Selection Technique*: Smart Touch, Land-on, Lift-off

Land-on is defined as the centroid of the ellipse that represents the first touch that contacts the screen. Lift-off is defined as the centroid of the ellipse that represents the last touch that contacts the screen. Land-on and lift-off are the selection schemes current touch systems use to determine if a target has been selected, as a user must land and lift within the bounds of their target. These techniques were treated as separate for the sake of measuring distance from an intended touch point (a crosshairs in this case). Area targets, such as buttons, were not used, as I was interested in understanding distance from target location, not hit rate.

My evaluation was conducted using a testing procedure based on methods used in machine learning [52] and followed in previous template-matching work [90]. For the given number of poses T performed by each participant (see Table 2), E poses were randomly extracted and used as training examples. E was increased from 10 to 40 in steps of 10. Of the remaining $T - E$ poses, one pose was randomly selected and treated as the candidate. The candidate was then matched against the library of E training examples (no candidate could serve as a template within the same trial). This procedure was performed 100 times for each participant per level of E for the first experiment. In total, 10 (participants) × 100 (trials) × 4 (E values) = 4000 matching tests were conducted. This procedure was also performed 100 times for each participant for the second experiment, resulting in a total of 10 (participants) × 100 (trials) = 1000 matching tests.

Error Offset, measured by the Euclidean distance between the center of the crosshairs in each trial and the produced (x, y) location reported by Smart Touch in the first experiment, and by each *Selection Technique* in the second experiment, was the dependent variable of interest. Error Offset was analyzed with a mixed-effects model analysis of variance [48]. My model used fixed-effects for the number of training examples E in the first experiment, for *Selection Technique* in the second experiment, and for Trial (nested within number of training examples) in both experiments.

3.4.2. Results

This section presents the results of my experiment to see which level of E provided the biggest increase in accuracy. I also present the results of my second experiment to see how well the best version of Smart Touch (found in the first experiment) could predict target locations compared to the touch sensors found in the PixelSense.

Mean distances from the intended target for each level of E from the first experiment are shown in Table 3. There was a significant effect of the number of training examples E ($F_{3,3591}$=4.53, p<.01) on Error Offset. To correct for multiple pairwise comparisons, we used Holm's sequential Bonferroni procedure [34]. Pairwise comparisons showed that E=30 was significantly more accurate than E=10 and E=20 (p<.05). Although E=30 was not significantly more accurate than E=40, we decided to use E=30 as the value for Smart Touch in the second experiment due to its lower average *Error Offset* compared to E=40.

Number of examples E	Smart Touch ***Error Offset*** **(cm)**
10	3.53 (3.65)
20	3.30 (3.69)
30	3.01 (3.31)
40	3.23 (4.39)

Table 3. Overall means for Error Offset (lower is better) for Smart Touch by number of training examples. Standard deviations are shown in parentheses.

In the second experiment, there was a significant effect of *Selection Technique* ($F_{2,2889}$=505.76, p<.0001) on *Error Offset*. Pairwise comparisons showed that Smart Touch (at E=30) was significantly more accurate than both Land-on and Lift-off (p<.0001). Smart Touch predicted (x, y) touch locations that were 30.71% and 28.26% of the distance to the target as the Land-on and Lift-off techniques, respectively. Put another way, Smart Touch predicted distances that were *over three times closer* to the intended targets than the native sensor techniques. In raw distance terms, this was 3.01 cm from the intended crosshairs for Smart Touch, and 9.80 cm and 10.65 cm for Land-on and Lift-off, respectively. Overall means and participant-specific error offsets are shown in Table 4.

3.6. Discussion

I wanted to discover whether Smart Touch could significantly improve touch accuracy for people with motor impairments. My experiments show that with an average offset error of 3.01 cm, Smart Touch was significantly better at predicting target locations compared to the land-on and lift-off locations reported by the touch sensors in the Microsoft PixelSense. Smart Touch was able to improve accuracy with only 30 templates, which could take under 8 minutes to collect. With these 30 templates, Smart Touch was over three times closer to intended target locations compared to land-on and lift-off. On average, Smart Touch was more accurate in predicting target locations for all 10 participants compared to land-on and lift-off. Although not all participants were able to complete all 110 trials due to fatigue,

	Error Offset (cm)		
ID	**Smart Touch (E=30)**	**Land-on**	**Lift-off**
1	6.39 (5.54)	26.15 (8.75)	35.34 (6.75)
2	2.71 (2.25)	7.33 (3.56)	6.94 (3.78)
3	3.88 (2.21)	9.39 (3.02)	8.47 (2.63)
4	1.04 (0.89)	1.67 (0.93)	1.96 (0.94)
5	1.73 (1.76)	7.87 (2.47)	7.99 (2.74)
6	5.09 (3.58)	17.07 (3.67)	16.50 (11.73)
7	1.02 (0.85)	1.15 (1.50)	1.75 (6.47)
8	1.77 (1.83)	5.24 (5.01)	4.21 (5.11)
9	2.21 (1.58)	7.14 (5.85)	7.91 (7.08)
10	4.29 (4.01)	14.98 (5.71)	15.49 (8.74)
MEAN	3.01 (3.31)	9.80 (8.59)	10.65 (11.41)

Table 4. Overall means for Error Offset by Selection Technique for each participant (lower is better). Standard deviations are shown in parentheses. Smart Touch was much better than Land-on and Lift-off at predicting the location of the participants' intended target.

I did not find any indication that fatigue negatively impacted the results of the experiment. In general, the touch behaviors of my participants remained consistent throughout all of the completed trials.

The average land-on and lift-off distances were 9.80 cm and 10.65 cm respectively. These distances are far greater than the widths of current touch screen widgets, which range from between 2.6 mm to 4.8 mm [62]. For most participants, even the recommended widget sizes for users with motor impairments (between 1.2 and 1.7 cm [29]) would not be big enough to accommodate their touch behaviors when using land-on or lift-off. With Smart Touch, however, increasing the widget size to 3cm would allow seven of my ten participants to more accurately select targets. Due to their large screen sizes, applications that run on interactive tables can afford to render larger targets.

My first experiment showed that Smart Touch can improve accuracy with only 30 templates. These templates could be collected in a standalone calibration procedure (similar to what was done in the exploratory study) in under 8 minutes. Templates could also be collected unobtrusively in the background. As tools and techniques for data collection "in the wild" are developed and deployed, Smart Touch could take advantage of these methods to collect many touch examples over an extended period of time.

3.6.1. Limitations

One limitation of this work is the generalizability of the touch behaviors exhibited by people with motor impairments to other touch screen devices, like phones or tablets. It is possible that the size and ergonomics of the PixelSense influenced the touch behaviors of my participants, although precautions were taken to ensure that any negative effects the PixelSense may have had on participants were reduced. Another limitation is the generalizability of Smart Touch to other touch screen devices. The PixelSense provides a wealth of touch data that is not easily available on all platforms. I do believe Smart Touch can achieve similar results on different devices, such as the Apple iPad, even if the reported touch data is not as extensive as the data reported by the PixelSense.

3.7. Summary

This chapter presented two contributions that further our understanding of how to improve touch input for people with motor impairments. First, I presented an exploration of the touch behaviors of ten people with motor impairments. From this exploration I identified two key behaviors that impact touch screen accessibility. One behavior is that users with motor impairments might touch the screen with multiple fingers or various parts of the hand, while system presumes that users interact with the screen using a single, fully extended finger. The second behavior is that users would use the screen for stability by dragging their hand along the screen towards their intended target. This behavior resulted in touch-down and touch-up locations that were far from the intended targets. I identified that it is important to consider a user's entire touch process, which includes all touches that occur from touch-down to the last touch-up.

From my observations during the exploratory study, I created Smart Touch, a three-step template matching algorithm that maps any arbitrary number of contact areas to the user's intended (x, y) touch location. Results from my evaluation of Smart Touch showed that Smart Touch could significantly improve touch accuracy over the land-on and lift-off locations reported by the Microsoft PixelSense with only 30 templates. Smart Touch was published at CHI 2016 and received a Best Paper Award (top 1% of all submissions) [57].

Chapter 4. Cluster Touch: Improving Smartphone Touch Accuracy*

This chapter presents an analysis of the touch offsets created by people with motor impairments when interacting with a smartphone touch screen. I also analyzed touch offsets produced by people without motor impairments while standing and while walking. This chapter also introduces Cluster Touch, a combined user-independent and user-specific touch offset model that improves smartphone touch screen accuracy for users with motor impairments with only 20 touch examples.

4.1. Motivation

Smartphones have increased in popularity over the past decade. According to the Pew Research Center, more adults in the United States own a smartphone than a tablet, laptop, or desktop computer[4]. Smartphones offer many useful benefits to users, like the ability to communicate with friends and family, complete work, and browse the internet while away from the home or office. However, many people with motor impairing conditions like cerebral palsy, muscular dystrophy, or Parkinson's disease, encounter accessibility challenges when interacting with smartphone touch screens [29,74]. Inaccurate touch input on smartphones can make it difficult or impossible to perform simple everyday tasks such as sending text messages, composing email, taking photos, or playing games.

The previous chapter explored the touch behaviors of people with motor impairments when interacting on a large touch screen device (Microsoft PixelSense interactive table). However, it is unknown how the touch behaviors of people with motor impairments are expressed on a smartphone touch screen. The larger form factor of the PixelSense may have allowed participants to express touch behaviors that are improbable to perform on a smartphone touch screen because of the smaller form factor. It is also unknown how accurately people without motor impairments, but who are under the effects of situational impairments [69,71], can touch various regions of a smartphone touch screen. People under the effects of situational impairments also experience difficulties when touching the

* Parts of this chapter are adapted from: Mott, M.E. and Wobbrock, J.O. 2019. Cluster Touch: Improving Touch Accuracy on Smartphones for People with Motor and Situational Impairments. *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '19). To appear.

4 http://www.pewresearch.org/fact-tank/2018/09/28/internet-social-media-use-and-device-ownership-in-u-s-have-plateaued-after-years-of-growth/

screen, which results in more input errors [6,27,40,47]. Understanding and correcting errors made by people with motor impairments, and people without impairments under the effects of situational impairments, will allow for more enjoyable and accessible experiences when interacting with smartphone touch screens.

Prior research efforts have investigated different machine learning based touch models to increase touch accuracy on smartphones [14,80]. Although machine learning models have been shown to improve touch accuracy, they typically require large amounts of data to achieve the biggest gains in accuracy. Performing hundreds or thousands of touch examples is problematic for people with motor impairments, as providing these examples requires significant time and effort on the part of the user. As I detailed in the previous chapter, fatigue often prevents users from contributing many touch examples in a short period of time. Thus, touch models aimed to improve touch accuracy for people with motor impairments must not require users to spend a significant amount of time providing training examples for the model.

This work makes three contributions towards expanding knowledge on the accessibility of smartphone touch screens for people with motor impairments. First, this work provides an examination of touch offsets generated by people with motor impairments, and people without motor impairments while standing and walking. Second, this work presents Cluster Touch, a combined user-specific touch model that corrects touch offsets on smartphones. Third, this work provides empirical results of two offline and one interactive evaluations of Cluster Touch.

4.2. Exploration of Touch Offsets

Touch input is often inaccurate because a finger occupies an entire area [77]. Touch is also inaccurate because participants have their own mental models of how their finger placement corresponds to their desired touch point on the screen [35,36]. When a user attempts to acquire a target, the distance between the location of the intended target and the reported touch location is a two-dimensional touch offset vector (Figure 10).

Prior work by researchers have investigated touch error offsets for users without impairments [14,32,35,36,80]. However, to the best of my knowledge, there have been no prior investigations into the touch error offsets created by people with motor impairments, or by people without motor impairments experiencing situational impairments. Without knowing how touch offsets impact accuracy for users with motor impairments, it will be difficult to devise solutions to correct for these

errors. To better understand the touch offsets created by people with motor impairments, I conducted a preliminary investigation where I collected touch data from users with and without impairments touching the center of crosshairs shown on a Google Nexus 6 smartphone.

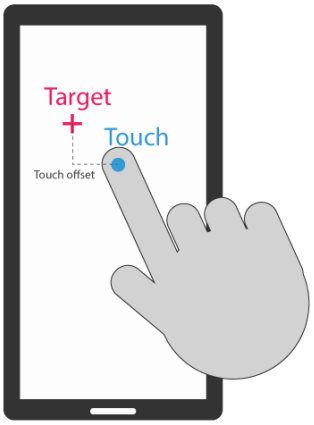

Figure 10. The difference between a user's touch location and the location of the intended target is a two-dimensional touch offset vector.

4.2.1. Participants

I recruited 4 people with motor impairments (3 female, 1 male, average age 31.5, *SD*=6.8), and 8 people without impairments (5 female, 3 male, average age 27.4, *SD*=5.1) to participate in my exploratory study. Participants with motor impairments were paid $30 USD and participants without motor impairments were paid $15 USD for their participation. Participants were recruited through word of mouth and through email listservs of organizations that support people with various motor impairments. All participants were right-handed. Two participants with motor impairments had cerebral palsy, one had muscular dystrophy, and one had multiple sclerosis.

4.2.2. Apparatus

Touch data was collected using an experiment testbed developed for Android 8.1 using C# and the Xamarin framework. The testbed captured and logged all touch events registered by the touch sensors in the Nexus 6. For each touch event, I recorded the reported x- and y-coordinates along with a timestamp. All sessions were conducted on a Google Nexus 6 smartphone (15.14 cm diagonal with a screen resolution of 1440 × 2560 pixels) running Android 8.1.

4.2.3. Procedure

Each participant completed a single lab session that lasted between 30 minutes to an hour. Participants with motor impairments completed 540 selection tasks while seated, while participants without motor

impairments completed 540 selection trials while standing and while walking on a treadmill. The treadmill was set to a speed that would allow participants to be able to walk comfortably while still being able to touch the screen effectively. Participants without motor impairments were instructed to hold the phone in their nondominant hand and to touch the screen with the index finger on their dominant hand. Participants with motor impairments were instructed to touch the screen with a finger on their dominant hand (Figure 11).

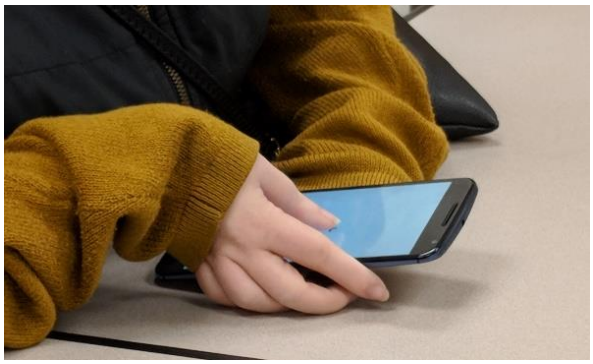

Figure 11. A participant with motor impairments completes a target selection trial on the Google Nexus 6 smartphone.

A single crosshairs was shown on the screen in each trial. Participants were instructed to touch the center of the crosshairs as accurately as possible. Crosshairs were equally spaced around the screen, but displayed randomly, to ensure that touch examples were collected from every screen region. No crosshairs was presented within 150 pixels of the screen border. A trial began when the first contact point was registered and was ended once all registered touches were lifted. At the end of each trial, a new crosshairs was displayed on the screen. Participants were not provided feedback on the location of detected touches. Feedback was removed to ensure that participants did not adjust their touch behaviors to appease the system. Instead, I wanted participants to touch naturally, without worrying about how the system interpreted their touches. Four (participants) × 540 (trials) = 2160 trials were collected from participants with motor impairments. Fifty-four outliers (touches that occurred more than 15 mm away from the intended target) were removed and not included in the analysis, resulting in a total of 2106 trials. Eight (participants) × 540 (trials) × 2 (postures) = 8640 trials were collected from participants without motor impairments.

4.2.4. Results

I collected a total of 2160 + 8640 = 10,800 trials from my 12 participants. In the following sections, I provide an analysis of the collected touch data. My analysis focused on two properties of touch error offsets, direction and magnitude.

4.2.4.1. Offset Direction

Offset direction refers to where touches occur relative to their intended targets. Prior investigations have found that the direction of touch offsets varies depending where on the screen users touch [14,32,80]. To analyze the touch offsets, I segmented the screen of the Nexus 6 into nine regions along its x-axis and fifteen regions along its y-axis. I chose this number of regions so that a suitable number of touch examples would be available for each segmented screen region. In each screen region I calculated the percentage of touches that occurred to the left and right of the intended crosshairs. I also calculated the percentage of touches that occurred above or below the intended crosshairs. Figure 12 shows a heat map of the offset directions along the x-axis (left or right) and along the y-axis (above or below).

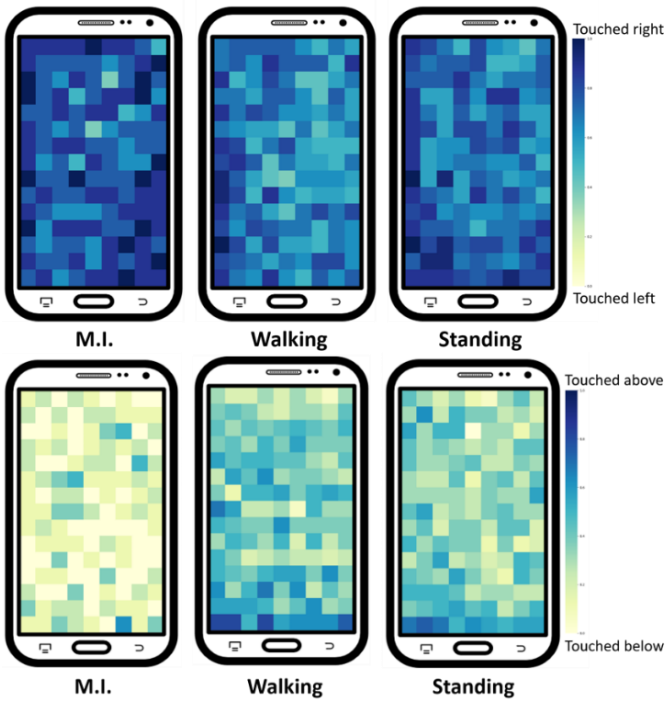

Figure 12. Heat maps of percentages of touches that occurred to the left or right of the intended crosshairs (top), and of touches that occurred below or above the intended target (bottom). "M.I." stands for users with motor impairments.

Participants in the three different user groups exhibited similar touch behaviors across various screen regions. As shown in the top of Figure 12, most participants touched to the right of the crosshairs. Overall, 83.3% of touches occurred to the right of the crosshairs for participants with motor impairments, and 63.7% and 69.7% of touches occurred to the right for users without impairments while walking and standing, respectively. Touch behaviors varied from the left to the right side of the screen. Targets presented on the left side of the screen resulted in more touches occurring to the right of the target, while targets presented on the right side of the screen resulted in more touches occurring to the left of the target. Across the three groups, crosshairs shown on the left-third region of the screen resulted in 35.2% of touches occurring to the left of the target, and crosshairs shown on the right-third of the screen resulted in 53.6% of touches occurring the left of the target.

With regards to the y-axis, most participants' touches were below the crosshairs. 84.6% of touches produced by participants with motor impairments were below the intended crosshairs, and 59.2% and 63.6% of touches occurred below the crosshairs for users without motor impairments while walking and standing, respectively. The bottom of Figure 12 shows that touches that occur above the intended crosshairs more frequently as crosshairs near the bottom of the screen. For the three groups, crosshairs shown on the top-third of the screen resulted in 31.7% of touches occurring above the target, and crosshairs shown on the bottom-third of the screen resulted in 42.5% of touches occurring above the target.

4.2.4.2. Offset Magnitude

The average distance between the first touch point and the intended crosshairs was 3.79 mm (*SD*=2.09) for participants with motor impairments, and 2.46 mm (*SD*=1.31) and 2.86 mm (*SD*=1.60) for participants without impairments while standing and walking, respectively. Similar to offset direction, offset magnitude tends vary depending on the screen region. I segmented the screen into nine regions along its x-axis and fifteen regions along its y-axis and calculated the average x- and y-offset distance for each region. Figure 13 shows a heat map of the average x-offset (top) and y-offset (bottom) error.

As shown in the top of Figure 13, the magnitude of x-offsets tends to be smaller on the right side of the screen compared to the left side. The average magnitude of x-offset errors was 2.26 mm (*SD*=1.91) for participants with motor impairments, and 1.53 mm (*SD*=1.15) and 1.76 mm (*SD*=1.38) for participants without impairments while standing and walking, respectively. Changes in magnitude

of y-offsets were less pronounced along the y-axis, but these errors accounted for a greater percentage of the total touch error. The average magnitude of y-offset errors was 2.70 mm (*SD*=1.64) for participants with motor impairments, and 1.61 mm (SD=1.23) and 1.89 mm (SD=1.60) for people without motor impairments while standing and walking, respectively.

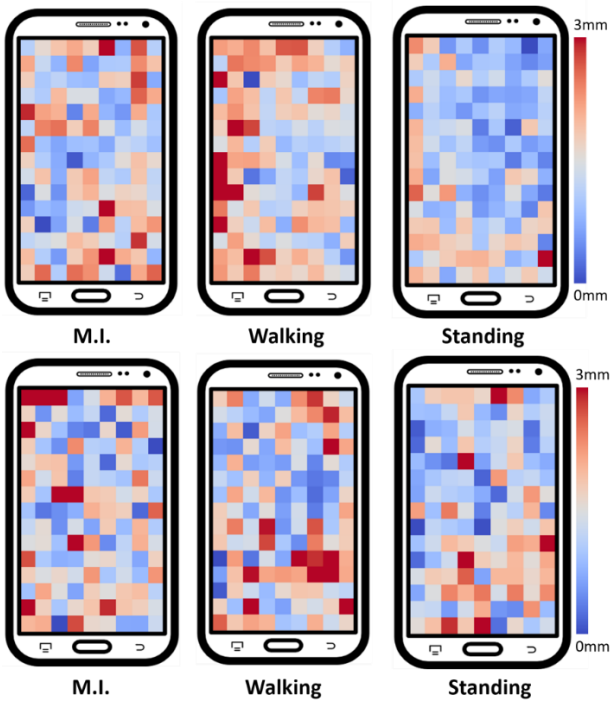

Figure 13. Heat maps of the average x-offset error (top) and average y-offset error (bottom). "M.I." stands for users with motor impairments.

4.2.5. Discussion

The data I collected during this investigation provided some useful insights into the touch behaviors of people with motor impairments, and people without motor impairments who were walking. Similar to previous investigations [32,80], my exploration of touch offsets found that touch offset direction varied depending where on the screen the user attempted to touch. The direction of the x- and y-offsets were actually quite similar across all three user groups, with participants typically touching below and to the right of their intended targets. Participants with motor impairments produced fewer touches that occurred to the left of targets compared to participants without motor impairments standing or walking. However, all groups exhibited the trend of producing more touches to the left of targets when they were placed on the right portion of the screen. It was also true that participants produced more touches above targets when they were placed on the lower portion of the screen.

The average touch error for participants with motor impairments was much higher than participants without motor impairments who were standing or walking. The y-offset error was the largest contributor of touch error for all three user groups, but the change in magnitude of y-offsets was less noticeable along the y-axis. The bottom of Figure 13 does show higher magnitude y-offset errors along the top-left portion of the screen for participants with motor impairments, which suggests that reaching towards this region of the screen was difficult for these participants. X-offset magnitudes were smaller along the right side of the screen compared to the left side for all three user groups.

These results show that the direction of touch offsets are similar across the three user groups, but the magnitude of offsets differs from group to group. Correcting these erroneousness offsets will allow people with motor impairments, and people without motor impairments under the effects of situational impairments such as walking, to have more successful experiences when engaging in everyday tasks such as texting or browsing the internet. Leveraging what I learned from my exploratory study of touch offsets, I created Cluster Touch, a combined user-independent and user-specific touch offset model that corrects touch offsets to provide more accurate touch predictions.

4.3. Design of Cluster Touch

My Cluster Touch model was designed to improve the accuracy of touch input on smartphone touch screens for people with motor impairments, and for people in motor impairing situations. A primary goal for Cluster Touch is to allow users to improve the accuracy of their smartphone touch screen quickly. Providing touch examples can require significant time and physical effort by people with motor impairments [57]. Because of the burden providing touch examples places on users, it important that Cluster Touch improve touch accuracy while requiring as little effort from the user as possible. As a result, Cluster Touch is a combined user-independent and user-specific touch model. The user-independent model captures common touch behaviors across multiple individuals. The user-independent model is then updated with touch examples provided by an individual user, allowing the model to adapt to a user's unique touch behavior. This section describes my user-independent and user-specific models.

4.3.1. User-Independent Model

From my exploration of touch offsets, I found that touch behaviors, such as touching to the right of crosshairs located on the left part of the screen, are similar across user groups. Cluster Touch takes advantage of this finding by combining touch examples from multiple users to create a user-

independent touch model. To build the user-independent model, I collected N touch examples from individual users. Each example consisted of the recorded touch location $\boldsymbol{t}$ and the location of the intended target $\boldsymbol{i}$, where $\boldsymbol{t}$ and $\boldsymbol{i}$ are two-dimensional screen coordinates. For each example, the x- and y-offsets $\boldsymbol{o_x}$ and $\boldsymbol{o_y}$ were computed by subtracting the location of the recorded touch from the location of the intended target (Equation 3):

$$\begin{aligned} \boldsymbol{o}_x &= \boldsymbol{i}_x - \boldsymbol{t}_x \\ \boldsymbol{o}_y &= \boldsymbol{i}_y - \boldsymbol{t}_y \end{aligned} \tag{3}$$

The screen was segmented into ten equal partitions along the x- and y-axes (*i.e.*, 10 columns for the x-dimension and 10 rows for the y-dimension). Each x- and y-offset was placed into one of these partitions depending on the location of $\boldsymbol{i}$. The average x-offset $\boldsymbol{o}_{\overline{x}}$ was computed for each partition along the x-axis, and the average y-offset $\boldsymbol{o}_{\overline{y}}$ was computed for each partition along the y-axis (Equation 4):

$$\begin{aligned} \boldsymbol{o}_{\overline{x}} &= \frac{\sum \boldsymbol{o}_x}{n} \\ \boldsymbol{o}_{\overline{y}} &= \frac{\sum \boldsymbol{o}_y}{n} \end{aligned} \tag{4}$$

Next, I used k-means clustering [31] on the averaged binned x- and y-offsets to identify screen regions where the magnitude of touch offsets were similar along the x- and y-dimensions of the screen (see Figure 14). From multiple iterations of testing, a k of 3 was chosen, which roughly creates clusters along the top, middle, and bottom of the screen in the y-dimension, and clusters along the left, middle, and right of the screen in the x-dimension. Because k-means was computed for the binned x- and y-offsets separately, our model consists of 6 total clusters. Each cluster $\boldsymbol{c}$ contains the location of the cluster in its respective dimension and a corrective offset. The benefit of clusters is that it allows for the generalization of touch behaviors to screen regions, removing the need to collect touch examples from every part of the screen. As a result, to update the user-independent model only a few examples from each cluster region are needed.

4.3.2. User-Specific Model

The user-independent model captures general smartphone touch behavior but is unaware of the touch abilities possessed by any specific user. My exploration of touch offsets showed that users' magnitudes vary between groups, with users with motor impairments exhibiting the greatest amount of error. Thus, the goal of the user-specific model is to keep the location of found clusters—as they represent

the locations where offsets tend to be similar—but to update the magnitude of the corrective offset of each cluster to better correct offsets produced by the given user.

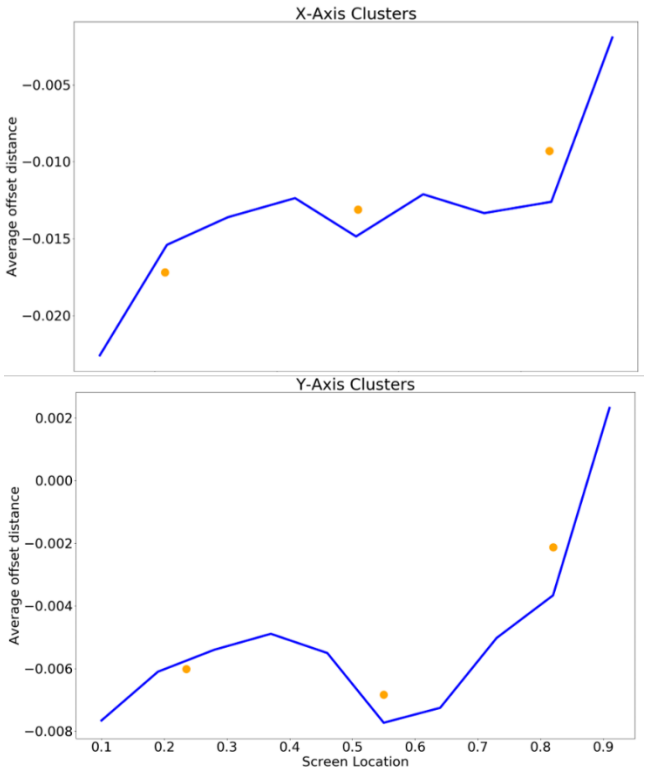

Figure 14. A plot of the average binned offsets along the screens' x- and y-axes. Cluster locations are shown in orange. Each cluster has a location (the value on the graphs' x-axes) and a corrective offset (the value on the graphs' y-axes).

To make the user-independent model user-specific, I collected touch samples from an individual user. For each sample, I calculated the *x*- and *y*-offsets (Equation 3) and binned them in their respective dimensions according to the proximity of $\boldsymbol{i}$ to the location of each cluster. There are three bins per dimension, one for each cluster. The average *x*- and *y*-offsets are then computed for each bin (Equation 4). Cluster offsets are updated by averaging the corrective offset for each cluster with the average offset in the bin closest to each cluster (Equation 5):

$$\begin{aligned} \boldsymbol{c'}_{\overline{x}} &= \frac{\boldsymbol{c}_{\overline{x}} + \boldsymbol{o}_{\overline{x}}}{2} \\ \boldsymbol{c'}_{\overline{y}} &= \frac{\boldsymbol{c}_{\overline{y}} + \boldsymbol{o}_{\overline{y}}}{2} \end{aligned} \tag{5}$$

I tested different weighted averages and found that averaging the binned and cluster offsets provided additional accuracy without introducing too much variance.

4.3.3. Touch Prediction

Given a new touch $\boldsymbol{t}$, I find clusters $\boldsymbol{c_x}$ and $\boldsymbol{c_y}$ that are closest to $\boldsymbol{t_x}$ and $\boldsymbol{t_y}$. If $\boldsymbol{t_x}$ or $\boldsymbol{t_y}$ are located between two clusters, we compute the interpolated corrective offset between the clusters depending on the location of $\boldsymbol{t}$ relative to the two clusters. If $\boldsymbol{t_x}$ or $\boldsymbol{t_y}$ are not located between clusters (*e.g.*, a touch occurs left of the leftmost cluster), the corrective offset of the closest cluster is taken. To predict the new touch point $\boldsymbol{t'}$, I add the corrective offsets $\boldsymbol{o}$—either interpolated or taken directly from a cluster—from each dimension to the given touch (Equation 6):

$$\begin{aligned} \boldsymbol{t}'_x &= \boldsymbol{t}_X + \boldsymbol{o}_X \\ \boldsymbol{t}'_y &= \boldsymbol{t}_y + \boldsymbol{o}_y \end{aligned} \tag{6}$$

4.3.4. Number of Training Examples

Because the goal of Cluster Touch is to allow users to quickly improve the accuracy of their touch screens, I tested different numbers of training examples to see how they impacted performance. I built a user-independent model from touch samples provided by participants in my exploratory study who were standing and created user-specific models with touch data from our participants with motor impairments. I incremented the number of samples E from 10 to 200 to see how well the user-dependent model performed with various numbers of training examples (Table 5).

Mean Error Distance (mm)						
E=0	E=10	E=20	E=50	E=100	E=150	E=200
2.84 (0.59)	2.61 (0.29)	2.57 (0.24)	2.55 (0.23)	2.54 (0.23)	2.54 (0.22)	2.53 (0.22)

Table 5. Mean error distance in millimeters (lower is better) for different numbers of training examples. Standard deviations are shown in parentheses.

Table 5 shows that increasing the number of training examples can improve touch accuracy. However, this increase in accuracy comes at the expense of time, which can be problematic for people with motor impairments. For my evaluation of Cluster Touch, we decided 20 training examples provided a good balance between speed (the time required to provide touch samples) and accuracy.

4.4. Evaluation of Cluster Touch

To determine how accurately Cluster Touch can predict users' intended touch locations, I performed an evaluation of Cluster Touch with 12 participants with motor impairments and 12 participants under the effects of situational impairments, namely walking. For my evaluation, I conducted three separate analyses. First, I evaluated Cluster Touch's ability to predict target locations during an interactive target selection task. Second, I performed an offline crosshairs analysis of Cluster Touch and compared its performance to two machine learning based touch offset models. Third, I performed a second offline

analysis to see if Cluster Touch could improve touch accuracy for existing mobile interfaces. For each study, I built a user-independent model using 1000 examples from participants in our exploratory study who were standing. Twenty touch examples were used to build the user-specific model.

4.4.1. Participants

I recruited 12 people with motor impairments (6 female, 6 male, average age 37.9 years, *SD*=13.7) and 12 people without impairments (7 female, 5 male, average age 24.8 years, *SD*=4.86) to participate in my study. Participants were recruited through the same means as described in the exploratory study. Participants with motor impairments were paid $30 USD and participants without impairments were paid $15. All participants with motor impairments were right-handed, and 11 participants without impairments were right-handed. Additional details about our participants with motor impairments can be found in Table 6.

				Self-reported impairments†										
ID	**Age**	**Sex**	**Health condition**	**Mo**	**Sp**	**St**	**Tr**	**Co**	**Fa**	**Gr**	**Ho**	**Se**	**Dir**	**Dis**
1	30	F	Cerebral Palsy	✓	✓	✓		✓	✓	✓	✓		✓	✓
2	21	M	Cerebral Palsy	✓	✓	✓		✓	✓	✓	✓		✓	✓
3	55	M	Multiple Sclerosis	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
4	19	F	Cerebral Palsy	✓		✓		✓	✓	✓	✓		✓	✓
5	38	F	Muscular Dystrophy	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓
6	43	F	Parkinson's Disease		✓	✓	✓	✓	✓	✓	✓		✓	✓
7	23	M	Cerebral Palsy	✓	✓	✓		✓	✓	✓	✓		✓	✓
8	39	F	Muscular Dystrophy	✓		✓		✓	✓	✓	✓		✓	✓
9	46	M	Multiple Sclerosis		✓	✓	✓	✓	✓	✓	✓		✓	✓
10	50	M	Cerebral Palsy	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
11	64	M	Parkinson's Disease	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓
12	27	F	Cerebral Palsy	✓		✓		✓	✓	✓	✓	✓	✓	✓

† Mo = slow movements, Sp = spasm, St = low strength, Tr = tremor, Co = poor coordination, Fa = rapid fatigue, Gr = difficulty gripping, Ho = difficulty holding, Se = lack of sensation, Dir = difficulty controlling direction, Dis = difficulty controlling distance.

Table 6. Demographic information for participants with motor impairments. Self-report categories are from Findlater et al. [20].

4.4.2. Apparatus

Touch data was collected using an experiment testbed developed for Android 8.1 using C# and the Xamarin framework. The testbed captured and logged all touch events. For each touch event, the

reported *x*- and *y*-coordinates of the touch were captured along with its timestamp. All sessions were conducted on a Google Nexus 6 smartphone (15.14 cm diagonal with a screen resolution of 1440 × 2560 pixels) running Android 8.1.

4.4.3. Procedure

Each participant completed a single lab session that lasted 30 minutes to an hour. Participants with motor impairments were asked to hold the phone in whichever way was most comfortable for them, and to touch the screen with a finger on their dominant hand. Participants without impairments were asked to hold the phone in their non-dominant hand and to touch the screen with a finger on their dominant hand. Each participant completed two tasks. First, each participant completed the crosshairs selection task described above in our exploration of touch offsets. Each participant completed 540 crosshairs selection trials.

Second, participants completed a target selection task where they were instructed to select a square 6×6 mm target placed in a grid as accurately as possible. A trial started when the participant touched the screen and ended when the finger was lifted. At the end of the trial, a new square in the grid was highlighted to indicate the next target. Successfully acquired targets were briefly highlighted green and unsuccessfully acquired targets were briefly highlighted red. Participants completed this task twice, once using the touch locations generated by the phone's touch sensors and again using predictions from Cluster Touch. A user-specific model was created for each participant using only 20 touch examples collected during the crosshairs task. Examples were selected to ensure they provided good coverage of the screen. Participants were unaware that Cluster Touch was active, and no feedback about touch locations were provided in either condition.

Participants with motor impairments completed both tasks while seated, and participants without impairments completed both tasks while standing and while walking. Twelve (participants) × 540 (trials) = 6480 crosshairs trials were completed by participants with motor impairments; and 12 (participants) × 540 (trials) × 2 (postures) = 12,960 crosshairs trials were completed by participants without impairments. Twelve (participants) × 50 (trials) × 2 (techniques) = 1200 interactive trials were completed by participants with motor impairments; and 12 (participants) × 50 (trials) × 2 (techniques) × 2 (postures) = 2400 interactive trials were completed by participants without impairments.

4.4.4. Design and Analysis

I completed separate analyses for data collected from participants with motor impairments and for data collected from participants without impairments. I conducted three experiments with each group.

The first was to determine if Cluster Touch could improve touch accuracy over the native touch sensor during an interactive target selection task. For my analysis of participants with motor impairments, my experiment was a within-subjects design with the following factor and levels:

- *Technique*: Nexus, Cluster Touch

My experiment for participants without motor impairments was a 2×2 within-subjects design with the same *Technique* factor as above, but with an additional factor:

- *Posture*: Standing, Walking

In all experiments, "*Nexus*" refers to the touch-down location reported by the native touch sensor on the Google Nexus 6.

My second experiment compared target prediction accuracy of Cluster Touch to the native touch sensor in the Nexus, and to two machine learning-based touch-offset models previously shown to improve touch accuracy [14,80]. Our analysis for participants with motor impairments was a within-subjects design with the following factor and levels:

- *Technique*: Nexus, Cluster Touch, Gaussian Process [80], Linear Offset [14]

The experiment for participants without motor impairments was a 4×2 within-subjects design with the same *Technique* factor as above and the same *Posture* factor as in the first experiment.

For this analysis, I implemented a testing procedure based on machine learning evaluations [52]. For each participant, I randomly selected 20 touch samples (*i.e.*, trials from the crosshairs selection task) as training examples for the three models in *Technique*. I used these models to predict intended touch points for the remaining trials. This procedure was performed 100 times for each participant, with a new set of training examples being randomly selected each time. I used the recommend parameters described in prior work [12] for the Gaussian Process [80] and Linear Offset [14] models. To train the models, I used the 1000 examples used to train the user-independent model as well as 20 randomly selected touch samples from each participant.

For my third experiment, I conducted an offline analysis of existing mobile user interfaces to determine if Cluster Touch could improve touch accuracy over the native touch sensor. I extracted 15,231 clickable targets from mobile interfaces in the RICO dataset [17]. For each crosshairs trial, I checked if the location of the crosshairs for that trial was within the bounds of a target extracted from the RICO dataset. Next, I used Cluster Touch to predict a new touch location for that trial. A "hit" occurred if the predicted touch landed within the bounds of the target. My analysis for participants with motor impairments was a within-subjects design with the same *Technique* factor as in the first experiment.

For participants without motor impairments, my experiment was a 2×2 within-subjects design with the same *Technique* factor as above and the same *Posture* factor as in the experiments above.

Target *Hit Rate*, the proportion of touches that successfully fell within the bounds of the target, was the dependent variable in the first and third experiments. The dependent variable of interest for the second experiment was *Error Distance*, measured by the Euclidean distance between the center of the crosshairs in each trial and the predicted target location of each technique. A mixed-effects model analysis of variance [48] was used to analyze *Error Distance*. Our model used fixed-effects for *Technique* and *Posture*. *Subject* was treated as a random effect to accommodate repeated measures. Pairwise comparisons were computed using Holm's sequential Bonferroni procedure [34] to correct for multiple pairwise comparisons. The nonparametric aligned rank transform procedure [33,64,84] was used to analyze *Hit Rate* for participants without motor impairments, as this measure did not conform to the assumptions of ANOVA. Wilcoxon signed-rank tests were used to analyze *Hit Rate* for participants with motor impairments.

4.5. Results

This section presents the results of our three experiments to determine the effectiveness of Cluster Touch for participants with and without motor impairments.

4.5.1. Interactive Selection Task

For participants with motor impairments, the average *Hit Rate* was 85.5% (*SD*=0.06%) with Cluster Touch and 74.5% (*SD*=0.08%) for Nexus. There was a significant main effect of *Technique* on *Hit Rate* (Z=-39.00, p<.001). The average *Hit Rate*s for participants without motor impairments are shown in Table 7. There was a significant main effect of *Technique* ($F_{1,33}$=73.69, p<.0001) and *Posture* on *Hit Rate* ($F_{1,33}$=58.05, p<.0001). There was no significant *Technique* × *Posture* interaction ($F_{1,33}$=0.63, *n.s.*).

Mean Hit Rate (%)

Technique	Posture	Hit Rate
Cluster Touch	Standing	95.33% (1.97%)
Nexus	Standing	91.67% (2.67%)
Cluster Touch	Walking	92.0% (3.07%)
Nexus	Walking	86.33% (3.70%)

Table 7. Overall means for Hit Rate (higher is better) for levels of Technique and Posture for participants without motor impairments. Standard deviations are in parentheses.

4.5.2. Crosshairs Prediction

Table 8 shows mean *Error Distance* for participants with and without motor impairments. There was a significant effect of *Technique* on *Error Distance* ($F_{3,33}$=109.34, p<.0001) for participants with motor impairments. Pairwise comparisons showed that Cluster Touch was significantly more accurate than Nexus, Gaussian Process, and Linear Offset (p<.01). For my group without impairments, there was a significant effect of *Technique* ($F_{3,77}$=7.57, p<.001) and *Posture* on *Error Distance* ($F_{3,77}$=168.93, p<.0001). There was no significant *Technique* × *Posture* interaction ($F_{3,77}$=0.58, *n.s.*). Pairwise comparisons showed that Cluster Touch was significantly more accurate than Nexus (p<.001) and Linear Offset (p<.05).

Mean Error Distance (mm)

Group	Cluster Touch	Gaussian Process	Linear Offset	Google Nexus 6
M.I.	2.81 (0.25)	3.17 (0.34)	3.15 (0.34)	3.42 (0.37)
Standing	2.22 (0.24)	2.31 (0.24)	2.31 (0.22)	2.41 (0.26)
Walking	2.67 (0.34)	2.80 (0.29)	2.80 (0.29)	2.85 (0.26)

Table 8. Overall means for Error Distance (lower is better) for levels of Technique and Posture for participants with and without motor impairments. Standard deviations are in parentheses.

4.5.3. RICO UI Analysis

For participants with motor impairments, the average *Hit Rate* was 90.0% (*SD*=8.30%) for Cluster Touch and 83.0% (*SD*=6.80%) for Nexus. There was a significant main effect of *Technique* on *Hit Rate* (*Z*=-39.00, p<.001). Average *Hit Rates* for participants without motor impairments are shown in Table 9. There was a significant main effect of *Technique* ($F_{1,33}$=82.45, p<.001) and *Posture* on *Hit Rate*. There was no significant *Technique* × *Posture* interaction ($F_{1,33}$=3.67, *n.s.*).

Mean Hit Rate (%)

Technique	Posture	Hit Rate
Cluster Touch	Standing	95.23% (1.85%)
Nexus	Standing	93.02% (2.01%)
Cluster Touch	Walking	93.22% (1.67%)
Nexus	Walking	89.08% (2.27%)

Table 9. Overall means for Hit Rate (higher is better) for levels of Technique and Posture. Standard deviations are shown in parentheses.

4.6. Discussion

The results from my experiments show that Cluster Touch can significantly improve smartphone touch screen accuracy for people with motor impairments and people in motor-impairing situations like walking. In the interactive selection tasks, Cluster Touch was 14.65% more accurate than the touch-down location reported by the Nexus 6's native sensor. These results are encouraging, as they demonstrate that Cluster Touch can be used in real time to improve smartphone touch accuracy. Furthermore, this increase in accuracy only required participants to provide 20 touch examples. Reducing the time and effort required to calibrate smartphone touch screens is an important goal, as users with motor impairments can struggle to complete a laborious calibration procedure. My offline analysis comparing Cluster Touch to the Gaussian Process [80] and Linear Offset [14] models showed that Cluster Touch can improve touch accuracy just as well or better than touch models based on machine learning when the number of user-provided examples is small. For participants with motor impairments, Cluster Touch was 11.36% more accurate than the Gaussian Process model and 10.79% more accurate than the Linear Offset model. Cluster Touch benefits from its combined user-independent and user-dependent touch models, as a greater percentage of user touch behavior is represented compared to shared models based on machine learning.

Although there were no significant *Technique* × *Posture* interaction effects for participants without motor impairments, Cluster Touch was on average more accurate for users while walking compared to the Nexus in the first experiment, increasing touch accuracy by 6.81%. Cluster Touch was also slightly more accurate in predicting target locations for walking participants compared to the Gaussian Process and Linear Offset models, improving accuracy by 4.64% over both models. These results demonstrate that Cluster Touch has the potential to provide benefits to users undergoing the effects of situational impairments like walking. Smartphones could detect when a user begins to walk and switch to the walking model, then switch back to the standing model when the user stops walking.

Cluster Touch also improved average touch accuracy for participants while standing by 4.37%, which suggests that this model could be used by a wide assortment of users or applications. For example, a mobile game developer might implement Cluster Touch to improve users' touch accuracy when playing their game.

My analysis of existing targets in the RICO dataset showed that Cluster Touch can improve touch accuracy for various real mobile applications. Of the more than 15,000 interface targets I tested, Cluster Touch improved touch accuracy by 8.21% over the Nexus 6 for participants with motor impairments. Improved touch accuracy of mobile applications would be a huge benefit to people with motor impairments, as more recreational and work-related applications transition from the desktop to smartphones.

4.6.1. Limitations

A limitation of this work is that I trained touch models for one pose, holding the phone in the non-dominant hand and touching the screen with the index finger from the dominant hand. It is possible that a model trained under these conditions may not perform as well in new situations. It is important to note, however, that my participants with motor impairments were not required to hold the device in any specific way, and Cluster Touch was still able to provide accuracy improvements for them. Further analysis of how different poses, such as one-handed use, impact model creation and performance is left for future work. Another limitation is that all but one of my study participants were right-handed, meaning that some of the touch behaviors I identified might be slightly different for left-handed users. Cluster Touch could build a left-handed user-independent model by collecting touch examples from several left-handed users. Other future work includes understanding how Cluster Touch might accommodate touch behavior change over time, as the touch abilities of people with motor impairments may fluctuate throughout the day for various reasons, such as fatigue or the effects of medication.

4.7. Summary

This chapter presented two contributions that further our understanding of how people with motor impairments, and people without motor impairments under the effects of situational impairments, touch on smartphone devices. First, I presented an analysis of the touch offsets produced by 4 people with motor impairments and 8 people without motor impairments. From this analysis I found that users with and without motor impairments exhibited similar touch behaviors regarding where touches

occurred relative to their intended targets. However, the magnitude of these touch offsets varied between the user groups, with participants with motor impairments exhibiting the greatest amount of error, followed by participants without motor impairments who were walking.

Based on my analysis of touch offsets, I created Cluster Touch, a combined user-independent and user-specific touch model that can improve touch accuracy with only 20 touch examples. In an evaluation of Cluster Touch, I found that Cluster Touch was significantly more accurate in predicting intended target locations for people with motor impairments compared to the touch sensors found the in Nexus 6 smartphone. Cluster Touch was also able to improve touch accuracy compared to two statistical machine learning models, Gaussian Process [80] and Linear Offset [14]. I also found that Cluster Touch was able to improve touch accuracy for people without impairments who were walking, which demonstrates that Cluster Touch has the potential to provide accuracy improvements for a range of users in different contexts.

Chapter 5. Exploring the Use of Convolutional Neural Networks to Predict Touch Locations

This chapter presents an early investigation into understanding how effective convolutional neural networks can be in predicting touch locations for people with motor impairments. In this chapter, I discuss implementation details to consider when using convolutional neural networks for touch prediction. I also provide results showing how accurately a trained convolutional neural network can predict touch locations from touch data collected in the studies described in Chapters 3 and 4. Finally, I conclude this chapter by identifying future research directions that can improve our understanding of how convolutional neural networks can be used for touch modeling and prediction.

5.1. Motivation

People with motor impairments employ different touch styles when interacting with touch screens. Depending on their abilities, people with motor impairments may touch with multiple fingers, the back of their hand, the side of their hand, or even their nose [2]. Other people with motor impairments may touch with a single finger, but they may not be as accurate as they would like due to muscle weakness or involuntary movements such as tremors [59,74].

My prior work described in Chapters 3 and 4 has investigated touch models to improve touch accuracy for people with motor impairments on different devices, and for users who exhibit different touch behaviors. Smart Touch was developed to accommodate varied multi-contact touches that unfold over time, and Cluster Touch was developed to accommodate single finger input on smartphone touch screens. These techniques, although shown to be useful in their respective contexts, were not designed to accommodate touch behaviors that are outside the constraints of their respective models. For example, Cluster Touch was not designed to accommodate the touch behaviors exhibited by participants in my Smart Touch study. Conversely, while Smart Touch could accommodate the touch behaviors of participants in the Cluster Touch study, it is doubtful that Smart Touch would significantly improve touch performance for these users, as Smart Touch does not take into account the variable touch behavior that occurs when touching a smartphone touch screen.

Because people with motor impairments exhibit many different types of touch behaviors—and these behaviors change depending on the form factor of the touch screen—it would be beneficial to have a single model that could accommodate a variety of touch behaviors. In an effort to create a

touch model capable of handling numerous touch behaviors, I investigated the practicality of using convolutional neural networks (CNNs) to build touch models that can accurately infer users' intended touch locations.

5.2. Why CNNs?

CNNs are neural networks primarily designed to classify and recognize objects within images [43,45]. CNNs are a member of a broader group of machine learning methods called deep learning [46]. CNNs typically consist of several layers: an input layer, convolutional layers, pooling layers, fully connected layers, and an output layer[5]. Each layer is responsible for performing different operations on the image. The convolution and pooling layers extract features from the image, and the fully connected layers handle classification. CNNs have been used for a variety of different applications, such as facial recognition [44], image classification [43], object detection [26], and natural language processing [7]. The popularity of CNNs has grown as the availability of digital images has increased, and as graphics processing units (GPUs) have gotten more powerful.

One of the benefits of CNNs is that they remove the need for extensive feature engineering. A feature is an attribute of collected data. For example, touch duration is an attribute inherently embedded in the data collected in experiments described in Chapters 3 and 4 that could be selected as a feature for a machine learning model. Feature engineering is the process of using domain expertise to identify which features of a dataset are the most important when building predictive models. The problem with feature engineering is that complex data can have numerous possible features, and it is not always clear which features will work best. As stated by Andrew Ng, cofounder of Google Brain, "Coming up with features is difficult, time-consuming, [and] requires expert knowledge."[6]

CNNs have not been previously used to build touch models, but using CNNs to model touch has potential benefits. The primary benefit of using CNNs to model touch is that it removes the need for feature engineering. The development of Smart Touch and Cluster Touch involved deciding which features to include in the models and what features should be excluded. Because the two models were developed to cater to users with different levels of motor control, each model places different emphases on certain touch features. For example, Smart Touch accommodates multiple finger input while Cluster Touch does not. Smart Touch also considers the size and orientation of touch contact

[5] http://slazebni.cs.illinois.edu/spring17/lec01_cnn_architectures.pdf
[6] https://forum.stanford.edu/events/2011/2011slides/plenary/2011plenaryNg.pdf

areas, while Cluster Touch only considers the (*x, y*) coordinate of the first reported touch. The touch models were also designed to accommodate touch behaviors on touch screens with different sizes. Screens with varying form factors may also require different feature considerations.

CNNs remove the need to try to determine which features are the most important for a particular group of users interacting on a particular device. Instead, touch data can be represented as images and fed into a CNN, which will learn through multiple training iterations the features of the images that are more salient. The benefit of this approach is that a single model could be built to improve touch accuracy for users who exhibit different touch behaviors interacting on devices with varying screen sizes. Thus, new models would not need to be created when attempting to improve touch accuracy on new devices, such as smartwatches.

5.3. Considerations when Building CNNs for Touch

Multiple approaches can be taken when building convolutional neural networks, depending on the task at hand and the data available. Because CNNs have not been previously used to model touch, it is unclear exactly how a CNN should be constructed to handle touch input, and how the touch data should be presented within the network.

5.3.1. Data Representation

Touch data is not typically presented as images, so it is important to consider how to represent the touch data. One possible representation is to use the raw data of a touch screen's capacitive sensors. Raw capacitive touch data provides access to the intensity of detected touches for each sensor. When represented as a grid, the raw capacitive image reveals the location of detected touches by representing the intensity of touches detected by the sensors as a different color than the background (e.g., contact areas would appear white on a black background). Another image representation would be to construct images from processed touch data. APIs for touch-based systems often report touches as ellipses, which approximate the capacitive values detected by the touch sensors. These ellipses can then be drawn on a canvas with the same dimensions as the touch screen from where the touches originated. In either case, all touches registered by the system at that point in time would appear in the image.

These two image representations have benefits and drawbacks. The benefit of the capacitive image is that it more accurately represents where contact areas are detected on the screen. As a result, there is more information present in the capacitive image compared to the processed image. Capacitive

images, however, are difficult to obtain, and not all touch systems enable access to their touch sensors' raw data. Processed images have the benefit of being more ubiquitous, as most touch systems represent on-screen touches as ellipses that can be easily converted to images. Processed images, however, do not contain as much information as capacitive images.

Another data representation to consider is whether to include a local or global view of the screen. Since CNNs analyze images at the pixel-level, images are commonly downsampled to ensure that the network is not trained on sparse data. For images of touch data, this is particularly important, as the area of interest typically occurs on a small region of a screen with a high resolution. For example, a touch image may include a single black ellipse located in the top-left corner of an otherwise all white screen background. A touch image with a local view would only show the screen region where the touch occurred, while the remaining portions of the image would be cropped out. Conversely, a global view would show the entire image.

There are benefits and drawbacks to each approach. The benefits of the local view are that the resolution of the image does not need to be downsampled, as cropping the area of interest would significantly reduce the size of the image. If using the global view, the image must be downsampled, as the image would be too large and too sparse for effective training. The local view does suffer from one flaw that I believe makes it unsuitable for touch modeling. Because the local view removes the empty regions of the screen (*i.e.*, regions of the screen where a touch did not occur), it also removes information regarding where the touch occurred on the screen. Knowing where touches occurred on the screen is important, as users' touch behaviors tend to change when targeting different regions of the screen. Results from the exploration of touch offsets in Chapter 4 showed that touch behaviors on the right portion of the screen differed from touches that occurred on the left portion of the screen. The same is true for touches that occurred at the top and bottom of the screen. Removing information that informs the model where the touch occurred will make it more difficult for the model to infer predicted touch locations, as the knowledge of how touch behaviors change depending on screen location would be lost.

5.3.2. Network Construction

In addition to considering how to represent the touch data, it is also important to consider how to construct the convolutional network. One possible approach is to use fully convolutional networks [49] (FCNs). FCNs are great for image segmentation tasks, where each pixel is assigned a label that represents the object the pixel is assigned to. The output of FCNs are images that represent the

segmented regions of the screen. For example, an FCN could be used to find the location of street signs in images. The network would take an image of a street as input and output an image where only the pixels that represented the street sign are active. For touch modeling, FCNs could be used to segment the screen region where it believes the user intended to touch. Touch images would be used as input and the output would be an image with pixels activated at the predicted touch location.

A problem with using FCNs for touch is that touch prediction often involves predicting the user's intended (*x, y*) touch location. Thus, the output of an FCN would need to be an image with a single pixel activated (the pixel would represent the intended (*x, y*) touch location). This use case is not well suited for FCNs, as the model may not select any pixels if it does not have high enough confidence to decide which pixel is the intended target.

A second network structure is a regular CNN. CNNs take an image as input and can perform regression or classification tasks. For touch modeling, CNNs would take touch images as input and would output an x-coordinate value and a y-coordinate value. CNNs are preferable to FCNs for touch modeling, as CNNs will always produce an x- and y-coordinate value.

5.4. Feasibility of CNNs for Touch Prediction

To understand how well CNNs can predict intended touch locations, I built a CNN and trained it using touch data collected from my Smart Touch investigation described in Chapter 3, and from touch data collected from my Cluster Touch investigation described in Chapter 4. For this exploration, I tried different setups to see how different data representations and model settings impacted performance. This investigation focuses primarily on data collected from the Smart Touch study, as this data has the greatest variability in image representation, and the average error for users in the Smart Touch study were quite high, which presents an opportunity for this CNN approach to provide accuracy improvements.

5.4.1. Setup 1

All CNNs were built in Python using Google's TensorFlow Machine Learning toolkit[7]. The CNN in the first setup was a regular convolutional neural network with 4 convolutional layers and 2 fully

[7] https://www.tensorflow.org/

connected layers. The model was user-specific, as the CNN was trained and evaluated using touch data from the same participant. Each users' model completed 200 training iterations.

To produce images for the model, I used the Smart Touch algorithm to extract the indicative poses from each trial performed by the 10 participants from my Smart Touch study. Each extracted contact area was displayed in black while the background was shown in white (see Figure 15). The dimensions of the images match the resolution of the Microsoft PixelSense, resulting in images with a 1920 × 1080 pixel resolution. The number of training examples for each participant is shown in Table 10. Participant 4 was excluded due to data processing issues.

All images were downsampled by a factor of three, resulting in a final image resolution of 640 × 360 pixels. As discussed previously, downsampling allows the model to take images that represent the full view of the screen.

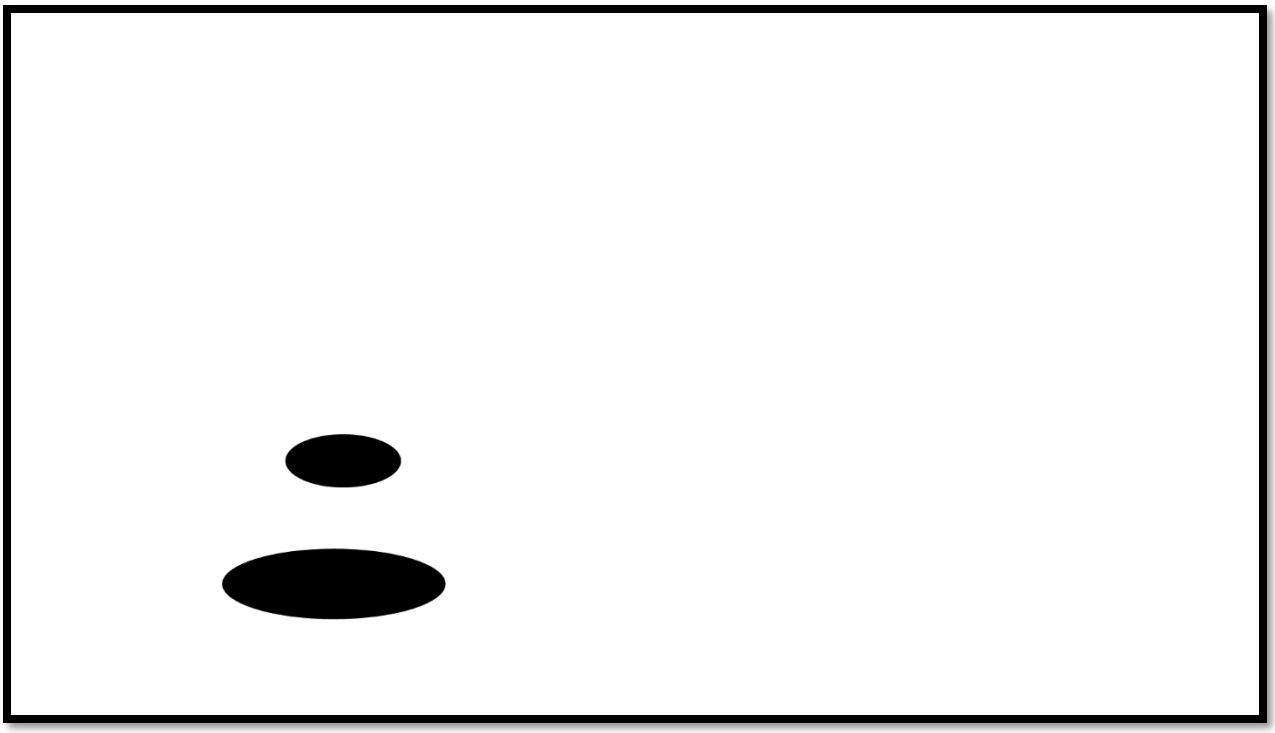

Figure 15. An example touch image from P1. The black ellipses represent the detected touch contact areas. The border is not present in the images used for training. The location of the crosshairs was encoded in the filename of each image.

The model was evaluated using 5-fold cross validation [52]. In 5-fold cross-validation, the data is randomly partitioned into 5 equally sized subsamples. During the evaluation, one of the 5 subsamples

is chosen as the test set while the remaining 4 subsamples are used to train the model. Each subsample is treated as the test set once.

5.4.2. Setup 1 Results

Results from Setup 1 show that the average error from the CNN model was higher than the average error from Smart Touch, as well as Land-on and Lift-off. The average error distance for the CNN, Smart Touch, and the land-on and lift-off locations reported by the PixelSense are shown in Table 10. The average error distance for the CNN was 14.94 cm, higher than Smart Touch (6.39 cm), and Land-on (9.80 cm) and Lift-off (10.65 cm).

		Error Offset (cm)			
PID	**Number of Training Examples**	**CNN**	**Smart Touch (E=30)**	**Land-on**	**Lift-off**
1	100	9.57	6.39	26.15	35.34
2	110	11.26	2.71	7.33	6.94
3	110	10.76	3.88	9.39	8.47
5	82	18.62	1.73	7.87	7.99
6	76	18.87	5.09	17.07	16.50
7	50	17.45	1.02	1.15	1.75
8	84	12.96	1.77	5.24	4.21
9	110	12.26	2.21	7,14	7.91
10	100	22.72	4.29	14.98	15.49
MEAN	91.3	14.94	3.01	9.80	10.65
SD	20.2	4.57	1.79	7.91	10.08

Table 10. Average error offset in centimeters for different selection techniques.

5.4.3. Setup 2

The second setup was a regular convolutional neural network with 3 convolutional layers and 2 fully connected layers. The model was user-specific, as the CNN was trained and evaluated using touch data from the same participant. Each users' model completed 100 training iterations. To produce images from my Smart Touch data, I used the same procedure described above. To produce images from my Cluster Touch data, the first contact area was shown in black on a white background (see

Figure 16). The dimensions of each image matched the resolution of the Google Nexus 6 smartphone, resulting in images with a resolution of 1440 x 2560 pixels.

In this setup, a local view was used instead of the global view used in Setup 1. The local view has the benefit of retaining more resolution about the touch, but information regarding where the touch occurred on the screen is lost. To create the local view, the largest bounding box which fits all of the contact areas in each trial is found for each participant. This bounding region is then used to crop the contact area(s) from the full image. The cropped images are then used to train the model.

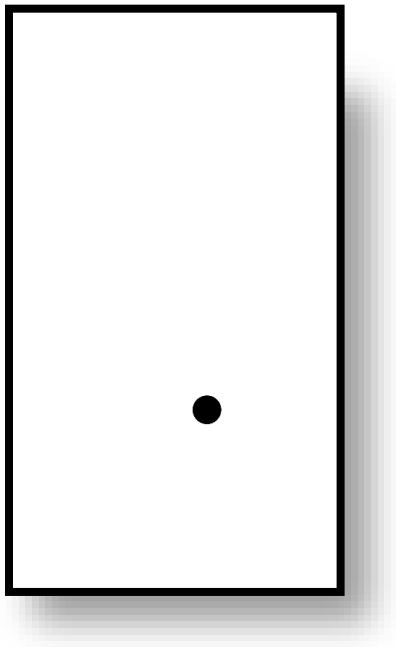

Figure 16. An example touch image from a participant from the Cluster Touch study. The black ellipses represent the detected touch contact areas. The border is not present in the images used for training. The location of the crosshairs was encoded in the filename of each image.

5.4.4. Setup 2 Results

For the Smart Touch data, results from Setup 2 showed that the average error from the CNN model (16.37 cm) was higher than the average error from Smart Touch (3.01 cm), Land-on (9.80 cm), and Lift-off (10.65 cm). The average error for the CNN model, Smart Touch, and the land-on and lift-off locations reported by the PixelSense are shown in Table 11. Results from Setup 2 also showed that for the Cluster Touch data, the average error from the CNN model (6.21 mm) was higher than the average error from Cluster Touch (2.81 mm) and the Nexus 6 (3.42 mm). The average error for the CNN model, Cluster Touch, and the Nexus 6 are shown in Table 12.

Error Offset (cm)					
PID	**Trials**	**CNN**	**Smart Touch (E=30)**	**Land-on**	**Lift-off**
1	100	10.44	6.39	26.15	35.34
2	110	12.67	2.71	7.33	6.94
3	110	12.13	3.88	9.39	8.47
5	82	21.53	1.73	7.87	7.99
6	76	20.26	5.09	17.07	16.50
7	50	19.62	1.02	1.15	1.75
8	84	13.96	1.77	5.24	4.21
9	110	12.59	2.21	7,14	7.91
10	100	24.14	4.29	14.98	15.49
MEAN	91.3	16.37	3.01	9.80	10.65
SD	20.2	5.00	1.79	7.91	10.08

Table 11. Average error offset in centimeters for different selection techniques for the Smart Touch data.

Error Offset (mm)			
PID	**CNN**	**Cluster Touch**	**Nexus 6**
1001	6.91	2.27	3.10
1002	6.09	2.60	3.66
1003	6.66	2.94	3.87
1004	7.16	2.65	3.05
1005	5.46	2.56	3.09
1006	4.98	2.58	3.67
1007	5.84	2.96	3.85
1008	5.79	2.64	3.15
1009	7.01	2.44	3.10
1010	5.40	2.62	3.60
1011	6.21	3.13	3.90
1012	5.55	2.64	3.01
MEAN	6.21	2.81	3.42
SD	5.55	0.25	0.37

Table 12. Average error offset in millimeters for different selection techniques for the Cluster Touch data.

5.5. Discussion

The results of this preliminary investigation showed that CNNs can be used to train touch models to predict touch locations. However, the models I built and tested were less accurate than the Smart Touch and Cluster Touch models described in Chapters 3 and 4. The CNN models were also less accurate than the land and lift locations reported by the Microsoft PixelSense, and the land locations reported by the Google Nexus 6 smartphone.

I tested two different setups, one that used the global image view and the other that used the cropped view. For the Smart Touch data, the local view in Setup 1 resulted in lower average error offsets compared to the global view in Setup 2 (14.94 cm compared to 16.37 cm). Although the local view in Setup 1 yielded slightly better results, it does not necessary mean that the local view is the most promising option. In the Smart Touch study, participants were asked to demonstrate the area on the screen where they felt comfortable touching, and targets only appeared in this region, often directly in front of the participant. As a result, participants did not have to touch areas all over the screen. Thus, in this specific case, the local view may be more beneficial, as behavior about targeting on different screen regions is not included in the dataset.

In real-world cases, however, using the global view may be more beneficial, because users exhibit different touch behaviors depending where on the screen they are targeting. Images that incorporate the global view may be able to capture these differences, as the hope is that the changes in targeting behavior will be learned by the model. One potential issue with the global view is that it might require more training examples than the local view. With the local view, the location of where the touch occurred is irrelevant, so each example can be thought of as a touch attempt on the same target. With the global view, each touch location is unique. As a result, the model did not receive multiple examples of the user touching in the same location, making it harder for the model to learn how to predict target locations for "new" touch points. When collecting data, it may help to collect two or three examples per touch point, which will allow the model to have multiple examples of how the participant touched in that particular screen region.

It is important to remember that Smart Touch's pose extraction algorithm was used to extract the indicative pose for each trial completed by my participants. It is unclear how well the model would have performed if images of the touch-down and touch-up contact areas would have been used instead. I presume that the model might have performed worse in those situations, as the touch-down

and touch-up locations exhibited by participants in my Smart Touch study were quite varied. This result has practical implications for how touch images should be constructed for model building. Although CNNs provide the benefit of not requiring feature engineering, the model still needs a touch for input. However, the theoretical question of what constitutes a "touch" remains. For Smart Touch, a "touch" comprised all touch events that occurred from the first touch-down event to the last touch-up event. Cluster Touch, however, only considered the initial touch-down event. CNNs can take a collection of touch images to build a touch model, but deciding which touches to generate images from remains an open question.

In Setup 2, I also tested the effectiveness of the model on the data from my Cluster Touch study. The trained model was less accurate than Cluster Touch and the land-on location reported by the Google Nexus 6. The Cluster Touch data presents a couple of challenges not present when using the Smart Touch data. The first is that the contact areas in the Cluster Touch data are much smaller and fewer in number compared to the contact areas in the Smart Touch data. Because the contact areas are so small initially, they get even smaller when downsampling the images, which means less resolution is available when using the global view. The second challenge is that the Cluster Touch data is composed of unique touch locations all over the screen. Because there are not multiple examples of touch instances, the model may have difficulty understanding how touch locations impact target locations.

The results of this preliminary investigation showed that CNNs can be used to train touch models. However, these models are not yet effective for accurately predicting users' intended touch locations. Further research is needed to understand how different data representations and model parameters impact the accuracy of CNNs for touch prediction

5.6. Future Research

I intend to continue conducting research in this domain. My future investigations will focus on three primary areas. The first is understanding how the number of training examples impacts the accuracy of the model. Deep learning systems are often trained with hundreds of thousands of data points. For the Smart Touch data, there was a maximum of 120 touch examples per participant, and some participants had few than 100 touch examples. For Cluster Touch, each participant had 540 training examples. Because 5-fold cross-validation was used, each model was only trained with 80% of the available data. Having more data available may allow the model to make improvements in touch

prediction. However, it is unclear how much data would be needed to see any gains in accuracy. To investigate this, I intend to conduct a study where I collect as much touch data as possible from people who exhibit similar touch behaviors to the participants from my Smart Touch investigation. I will focus on this group of users because improvements in touch accuracy will help these users the most. Collecting additional touch data will allow me to study how increasing the number of training examples impacts touch accuracy. These results will have practical implications for the use of CNNs for touch prediction. If thousands of touch examples are needed in order to see accuracy improvements from the model, then CNNs may be impractical for people with motor impairments, as supplying thousands of touch examples would require significant time and physical effort.

My second area of focus will be to investigate how capacitive images impact model performance. Although capacitive images are harder to acquire, they do contain more information, which might help the model during training. I intend to build an experiment testbed that can capture the value of the capacitive sensors as well as the processed touches reported by touch APIs. I will then build and train different models using the two sources of data to determine if either data representation yields accurate predicted touch locations.

The third focus area will be to understand how changing model parameters impacts performance. For example, increasing the number of convolutional layers would allow the model to handle more complexity. However, increasing the number of layers means the model must learn more parameters, which often means the model would require more training data. By collecting additional training data, I can systematically change model parameters to see how they impact performance.

5.7. Summary

This chapter presented an investigation into the feasibility of using CNNs to construct touch models. I discussed implementation details to consider when using convolutional neural networks for touch prediction. I also provided results showing how accurately a trained convolutional neural network could predict touch locations from touch data collected from my Smart Touch and Cluster Touch studies. The results showed that the CNN model was able to make touch predictions, but that these predictions were less accurate than native touch sensors and my two ability-based touch models. This exploration provides insights into how CNNs may be constructed, and how touch data may be represented, in order to build touch models using CNNs. Improving upon the results of this preliminary investigation remains an important priority for my future work.

Chapter 6. Discussion

In this chapter, I discuss practical and theoretical implications and limitations of the work presented in this dissertation. The topics discussed are: (1) what is a "touch;" (2) calibration procedures for touch screens; (3) accounting for changes in ability; (4) privacy implications of touch models; and (5) the transferability of touch models.

6.1. What is a Touch?

In my investigations of touch behaviors exhibited by people with and without motor impairments, one key question was present in each situation: what is considered a "touch?" Determining what constitutes a touch is an important question when constructing touch models. In my Smart Touch study, described in Chapter 3, the touch behaviors exhibited by my participants led me to the realization that a touch is not a simple action with two key events, down and up. Instead, a touch encompasses all the events that occur between touch-down and touch-up. Also, it cannot be assumed that all touches will be performed by a single finger.

Thus, it is important to consider what will and will not be considered a touch when constructing touch models, as it cannot be assumed that a "touch" represents any one particular action or event. How a "touch" is defined has practical implications when attempting to understand how touch models might work in real user interfaces. For example, many participants in my Smart Touch study dragged their hand across the screen as they approached their target. In a real user interface, this dragging action might be classified as a swipe or another multitouch gesture. Smart Touch, however, considers all touch events to be part of the user's touch process. This can be problematic, as Smart Touch does not currently have a way to distinguish between motion-based gestures and touches.

Also, because Smart Touch does not predict a touch location until the final touch-up event, using Smart Touch in real interfaces would make it difficult to perform actions such as drag-and-drop and a long-press. It is possible that Smart Touch could be augmented with new features that would allow users to perform these various actions while still considering the user's touch process. For example, Smart Touch could make predictions about the user's intended touch location throughout the touch process and provide visual feedback to the user regarding the predicted touch location. Then, Smart Touch could use a dwell-based technique to allow users to indicate that they want to perform secondary actions on the target.

Unlike Smart Touch, Cluster Touch considers a touch to be the first contact made with the screen. This method is more closely aligned with how current touch systems operate, but this approach would not be suitable for many participants from my Smart Touch study, as accurately landing near their intended target without prematurely touching the screen would be difficult.

6.2. Calibration Procedures for Touch Screens

Calibration procedures are common, and often necessary, when interacting with different technologies. For example, eye trackers often require users to calibrate the system before use [65]. Speech recognition systems may also require the user to calibrate, often by repeating a standard set of words or phrases. Although there have been resistive touch screens meant for use with styli that commonly employed quick calibration procedures upon startup, everyday capacitive touch screens typically do not require such procedures. The lack of calibration procedures is partially because touch input is quite accurate and intuitive for many people without disabilities. However, this dissertation showed that calibration procedures could be used to collect touch examples to feed into the different touch models.

Although calibration procedures were used, no consistent and reliable calibration procedure was established. There are two open questions related to how calibration procedures can be developed for touch screens. The first, and most important, is how many touch examples must users provide during calibration. There is currently no clear answer to this question, as the number of training examples needed relies heavily on the touch model being trained and the consistency of the touches provided by the user. For example, Cluster Touch, and models built with convolutional neural networks (presumably) would benefit from larger amounts of touch data, as the models are attempting to find patterns across all trials. Smart Touch, however, only ever maps candidate touch pattern to a single template, so while increasing the number of training examples provides additional templates for matching, no patterns are being learned from the aggregate data.

Fatigue and physical effort must also be kept in mind when considering the number of training examples needed for calibration. Participants in my Smart Touch study struggled to perform over 110 trials while participants in my Cluster Touch study were able to complete over 500 trials in a single setting. Requiring fewer training examples would alleviate some of the burden placed on the user, but fewer training examples often means reduced accuracy. Thus, it is important to find a nice mix of both

speed (regarding calibration time) and accuracy (regarding model performance). Cluster Touch negotiated this tradeoff by opting for fewer training examples at the expense of accuracy.

The second question is where should training targets appear on the screen. The calibration procedure should collect touch examples from across all screen regions, as the model should not contain any "blind spots." However, more specific calibration procedures can be enforced to improve accuracy of certain applications. For example, WalkType [27] was built to improve mobile text entry, so the model was built only using touches that occurred on the on-screen keyboard. It is important for researchers to consider these factors when developing touch screen calibration procedures.

6.3. Accounting for Changes in Ability

One limitation of the work presented in this dissertation is that touch models are trained and tested during the same lab setting. It is unknown whether the models trained during the experiment would provide the same benefits to the user a day, a week, or a month after the model was trained. Understanding changes in ability is particularly important for users with motor impairments, as their abilities may fluctuate due to fatigue or medication. Thus, it cannot be guaranteed that the touch behavior exhibited by a participant with a motor impairment in one study session will match the touch behavior that same person exhibits in another study session. Ability also changes according to context, so it is also possible that the touch behaviors exhibited in the lab will not match behaviors users exhibit while lying in bed or sitting on the couch.

Understanding how touch abilities change over time would provide useful insights for determining how best to update touch models. One possible way to understand how touches change over time is to use experience sampling [15]. An application on the user's device could periodically ask the user to input several touch examples and to answer a couple of quick questions, such as if they feel fatigued or if they took any medication recently. Another approach would be to gather touch information unobtrusively in the background. This method would not require any direct intervention by the user, but constructing applications that collect data discretely in the background would be difficult for a couple of reasons. First, touch models require ground truth data, which means that the model must have the touch data and the location of where the user was intending to touch. Inferring users' intended touch locations is difficult, as the application would have trouble identifying which touches were intentional and which touches were errors. The exact (x, y) coordinate of the intended

touch location would also be difficult to infer, as the system would not know where precisely the user is targeting.

Overall, it is important for touch models to consider how users' abilities might change over time and in different contexts. Touch models that can be updated and refreshed to match the abilities of the user at a given time might have important practical benefits on how touch models are adopted and used in real-world scenarios.

6.4. Privacy Implications of Touch models

Touch models contain implicit information about the abilities of users. This information could be potentially used to discriminate against participants with motor impairments. Recent work by Hamidi *et al.* [30] found that people who use assistive technologies have concerns about how the data that is used to build adaptive user interfaces may also be used to violate their privacy. For example, an employee may not want their boss to know that they use adaptive software at work. Further, the employee would not want their boss to have access to the data they provided to the adaptive software, as this information might contain details about the severity of their impairment.

Touch models are susceptible to these kinds of privacy incursions. Even if the data used to build the models was not available, the output of the models may be enough to indicate the level of motor control possessed by the user. For example, if a manager temporarily borrowed a tablet with an active touch model, the manager might be able to infer how well her employee can touch based on the touch feedback provided by the system (*e.g.*, touching in one location but the feedback suggested that the person intended to touch somewhere else). These privacy implications are important and must be considered when developing touch models for people with motor impairments.

6.5. Transferability of Touch Models

This dissertation investigated two touch models, Smart Touch and Cluster Touch, and a deep-learning model based on convolutional neural networks that were built to accommodate touch behaviors on different touch screen devices. One important question to consider is whether a touch model built on one device can be transferred to other devices with different screen sizes and resolutions. Prior work has shown that touch models can be transferred to different devices [14]. However, that work was not conducted with participants with motor impairments, so it is unclear whether touch models created for people with motor impairments can be transferred to different devices. One aspect that is important to consider is that changing the form factor of the device may cause users with motor

impairments to alter their touch behaviors. For example, if a person with a motor impairment uses a smaller form factor smartphone because it is easier to hold, the touch behaviors he exhibits on that device may not be the same as the behaviors he exhibits on a larger smartphone. Thus, the model trained on the smaller phone would not be applicable on the larger phone.

Understanding whether and how touch models trained for people with motor impairments can transfer to different devices is an interesting research question. Smart Touch models can be trained on different devices, but I do not believe that a Smart Touch model trained on one device would transfer easily to a different device with a different screen size. Because Smart Touch considers the user's entire touch process, templates generated on one screen size may be quite different than the templates that can be generated on another screen. For example, participants in my Smart Touch study used the screen of the PixelSense for stability as they approached their targets, and the larger screen allowed participants to utilize various parts of their hands without occluding the entire screen. These behaviors would not transfer well to the Google Nexus 6 device used by participants in the Cluster Touch study. Cluster Touch has greater potential to be transferable to different devices, but differences in touch behaviors of different sized devices would still impact performance. Further research is needed to understand how people with motor impairments express touches on touch screens of varying size.

Chapter 7. Future Work and Conclusion

In this chapter, I discuss possible directions for future work and summarize the contributions of this dissertation.

7.1. Future Work

I touched on some areas for future work in the previous chapter. Here, I will discuss two other possible research directions: (1) accessible gesture interaction for people with motor impairments; and (2) improving the accessibility of public touch screens.

7.1.1. Accessible Gesture Interaction for People with Motor Impairments

The work I presented in this dissertation has focused solely on improving touch input for people with motor impairments. Touch, however, is only one way users interact with touch screen devices. Gestures are a common interaction method, but is it unknown how accessible common touch screen gestures are for people with motor impairments.

One possible approach would be to mimic the gesture elicitation [83,88] study Kane *et al.* [39] conducted to understand the accessibility of gestures for blind people. Participants could be shown an action and then be asked to perform a gesture that would trigger that action. This study would provide useful information on how well users with motor impairments can perform certain gestures. It would also provide useful information on which gestures participants would like to perform in order to activate certain commands. It might be the case that a participant has difficulty performing certain gestures and would prefer to use new gestures to activate those commands. The gesture data could also be analyzed using various gesture recognizers [3,75,90] to see how well gestures made by people with motor impairments could be recognized and classified.

Another area of research regarding gesture interaction includes improving classifiers so that slips (*i.e.*, when a user's finger contacts the screen but slides shortly after touching) are identified and treated as a touch and not as a gesture. Slippage is common for users who exhibit tremors [53,74]. Understanding how a user performs a swipe gesture and how that gesture compares to accidental slips could improve touch screen accessibility for people with motor impairments.

7.1.2. Improving the Accessibility of Public Touch Screens

My dissertation focused on improving the accessibility of personal touch screen devices. However, as noted in the Introduction, touch screens are prevalent in many public spaces, such as airports and

grocery stores. It is unclear, however, how we may go about improving the accessibility of public touch screens for people with motor impairments. Touch models are a convenient way to improve touch accuracy on personal devices, as it can be assumed that the person interacting with the device is most likely the owner. There are, of course, exceptions, such as when a device is shared between multiple people in a household.

Public touch screens pose a significant challenge because touch models cannot be easily transferred to public devices. One possible option would be to allow users to quickly calibrate the touch screen before use. In situations where users interact with the public touch screen for an extended period of time, such as when voting, calibration may be beneficial. My Cluster Touch study showed that improvement in touch accuracy can be achieved with as few as 20 touch examples. Depending on the touch abilities of the user, it may only take a minute or two to collect 20 touch examples. However, in situations where the interaction with the screen is brief, it is impractical to ask users to spend time and effort calibrating the screen.

Public touch screens are particularly inaccessible to many people with neurodegenerative diseases such as ALS who have lost control of their arms or hands. In these situations, different measures may be required. One possible solution would be to attach a robotic arm with an embedded camera module[8] to the user's wheelchair. The camera view could then be transposed onto another device, such as a tablet mounted to the wheelchair. Users could then use gaze, voice, or another input method to indicate where they would like the robot arm to touch. This approach may allow people in wheelchairs to have more accessible interactions with public touch screens, such as ATMs or self-help kiosks at airports.

7.2. Conclusion

Touch input is a popular form of interaction for modern computing systems, but it still poses significant accessibility challenges to many people with motor impairments. This dissertation has demonstrated that ability-based touch models can significantly improve touch accuracy for people with motor impairments when interacting with interactive tables and smartphone touch screens.

I have described how people with motor impairments touch on interactive tabletops. The findings from this investigation showed that people with motor impairments exhibit two behaviors

[8] https://www.ufactory.cc/#/en/uarmswift

that impact touch screen accessibility. First, participants used the screen for stability by landing far from their intended targets and dragging their hands along the screen towards their targets. Second, participants would touch with various parts of their hands or multiple fingers. As a result, the system would register multiple concurrent touch points, making it impossible for the system to infer where the user was intending to touch. Based on these observations, I built and evaluated Smart Touch, a user-specific touch model that can resolve any number of arbitrary touch points to predict the user's intended (*x, y*) target location.

I have also described how people with motor impairments touch on a smartphone touch screen. From this investigation, I found that the touch offsets produced by people with motor impairments were similar in direction to the touch offsets produced by people without motor impairments who were standing and walking. I also found that although the direction of offsets was similar between the user groups, the magnitude of the errors were different, with people with motor impairments exhibited more errors than participants in the other groups. From these observations, I built and evaluated Cluster Touch, a combined user-specific and user-independent touch model that improves touch accuracy of smartphone touch screens with only 20 touch examples.

In addition to the touch models described above, I also investigated the feasibility of using convolutional neural networks to build a touch model capable of predicting intended target locations. In a preliminary investigation using touch data from participants from my Smart Touch and Cluster Touch studies, I found that CNNs can be trained to predict touch locations, but that these predicted touch locations are less accurate than the touch locations provided by my two ability-based touch models, and the native touch sensors found in the Microsoft PixelSense and the Google Nexus 6. This exploration provides useful details on what to consider when representing touch data and choosing model parameters for building a CNN capable of modeling human touch behavior.

In presenting this research, this dissertation has demonstrated the following thesis:

Ability-based touch models can improve touch accuracy on touch screens compared to native touch sensors or existing statistical models for people with motor impairments and for people in motor-impairing situations.

My work has shown that touch models designed to accommodate the touch behaviors of people with motor impairments can improve touch accuracy. However, more work is needed to understand how to update touch models to accommodate ability changes over time, how to create robust and reusable touch screen calibration procedures, and how to improve the accessibility of

public touch screen displays. I hope that this dissertation will motivate future researchers to explore how to continue improving the accessibility of touch screens for users with any and all levels of ability.

References

1. Pär-Anders Albinsson and Shumin Zhai. 2003. High precision touch screen interaction. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '03), 105–112. https://doi.org/10.1145/642611.642631
2. Lisa Anthony, YooJin Kim, and Leah Findlater. 2013. Analyzing user-generated Youtube videos to understand touchscreen use by people with motor impairments. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '13), 1223–1232. https://doi.org/10.1145/2470654.2466158
3. Lisa Anthony and Jacob O. Wobbrock. 2012. $N-Protractor: A fast and accurate multistroke recognizer. In *Proceedings of Graphics Interface 2012* (GI '12), 117–120. Retrieved April 2, 2017 from http://dl.acm.org/citation.cfm?id=2305276.2305296
4. Ravin Balakrishnan. 2004. "Beating" Fitts' law: Virtual enhancements for pointing facilitation. *International Journal of Human-Computer Studies* 61, 6: 857–874. https://doi.org/10.1016/j.ijhcs.2004.09.002
5. Hrvoje Benko, Andrew D. Wilson, and Patrick Baudisch. 2006. Precise selection techniques for multi-touch screens. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '06), 1263–1272. https://doi.org/10.1145/1124772.1124963
6. Joanna Bergstrom-Lehtovirta, Antti Oulasvirta, and Stephen Brewster. 2011. The effects of walking speed on target acquisition on a touchscreen interface. In *Proceedings of the ACM Conference on Human Computer Interaction with Mobile Devices and Services* (MobileHCI '11), 143–146. https://doi.org/10.1145/2037373.2037396
7. Ashwin Bhandare, Maithili Bhide, Pranav Gokhale, and Rohan Chandavarkar. 2016. Applications of convolutional neural networks. *International Journal of Computer Science and Information Technologies* 7, 5: 2206–2215.
8. Xiaojun Bi, Yang Li, and Shumin Zhai. 2013. FFitts law: Modeling finger touch with Fitts' law. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '13), 1363–1372. https://doi.org/10.1145/2470654.2466180
9. Xiaojun Bi and Shumin Zhai. 2013. Bayesian touch: A statistical criterion of target selection with finger touch. In *Proceedings of the ACM Symposium on User Interface Software and Technology* (UIST '13), 51–60. https://doi.org/10.1145/2501988.2502058
10. Xiaojun Bi and Shumin Zhai. 2016. Predicting finger-touch accuracy based on the dual gaussian distribution model. In *Proceedings of the ACM Symposium on User Interface Software and Technology* (UIST '16), 313–319. https://doi.org/10.1145/2984511.2984546
11. Pradipta Biswas and Patrick Langdon. 2012. Developing multimodal adaptation algorithm for mobility impaired users by evaluating their hand strength. *International Journal of Human-Computer Interaction* 28, 9: 576–596. https://doi.org/10.1080/10447318.2011.636294
12. Daniel Buschek and Florian Alt. 2015. TouchML: A Machine learning toolkit for modelling spatial touch targeting behaviour. In *Proceedings of the International Conference on Intelligent User Interfaces* (IUI '15), 110–114. https://doi.org/10.1145/2678025.2701381
13. Daniel Buschek and Florian Alt. 2017. ProbUI: Generalising touch target representations to enable declarative gesture definition for probabilistic GUIs. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '17), 4640–4653. https://doi.org/10.1145/3025453.3025502
14. Daniel Buschek, Simon Rogers, and Roderick Murray-Smith. 2013. User-specific touch models in a cross-device context. In *Proceedings of the ACM Conference on Human-Computer Interaction with Mobile Devices and Services* (MobileHCI '13), 382–391. https://doi.org/10.1145/2493190.2493206

15. Sunny Consolvo and Miriam Walker. 2003. Using the Experience Sampling Method to Evaluate Ubicomp Applications. *IEEE Pervasive Computing* 2: 24–31.
16. Andrew Crossan, Roderick Murray-Smith, Stephen Brewster, James Kelly, and Bojan Musizza. 2005. Gait phase effects in mobile interaction. In *Extended Abstracts on Human Factors in Computing Systems* (CHI EA '05), 1312–1315. https://doi.org/10.1145/1056808.1056904
17. Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. 2017. RICO: A mobile app dataset for building data-driven design applications. In *Proceedings of the ACM Symposium on User Interface Software and Technology* (UIST '17), 845–854. https://doi.org/10.1145/3126594.3126651
18. Morgan Dixon, James Fogarty, and Jacob O. Wobbrock. 2012. A general-purpose target-aware pointing enhancement using pixel-level analysis of graphical interfaces. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '12), 3167–3176. https://doi.org/10.1145/2207676.2208734
19. Sacha N. Duff, Curt B. Irwin, Jennifer L. Skye, Mary E. Sesto, and Douglas A. Wiegmann. 2010. The effect of disability and approach on touch screen performance during a number entry task. In *Proceedings of the Human Factors and Ergonomics Society* (HFES '10), 566–570.
20. Leah Findlater, Alex Jansen, Kristen Shinohara, Morgan Dixon, Peter Kamb, Joshua Rakita, and Jacob O. Wobbrock. 2010. Enhanced area cursors: Reducing fine pointing demands for people with motor impairments. In *Proceedings of the ACM Symposium on User Interface Software and Technology* (UIST '10), 153–162. https://doi.org/10.1145/1866029.1866055
21. Leah Findlater, Karyn Moffatt, Jon E. Froehlich, Meethu Malu, and Joan Zhang. 2017. Comparing touchscreen and mouse input performance by people with and without upper body motor impairments. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '17), 6056–6061. https://doi.org/10.1145/3025453.3025603
22. Jon Froehlich, Jacob O. Wobbrock, and Shaun K. Kane. 2007. Barrier pointing: Using physical edges to assist target acquisition on mobile device touch screens. In *Proceedings of the ACM SIGACCESS Conference on Computers and Accessibility* (Assets '07), 19–26. https://doi.org/10.1145/1296843.1296849
23. Krzysztof Gajos and Daniel S. Weld. 2004. SUPPLE: Automatically generating user interfaces. In *Proceedings of the ACM Conference on Intelligent User Interfaces* (IUI '04), 93–100. https://doi.org/10.1145/964442.964461
24. Krzysztof Z. Gajos, Daniel S. Weld, and Jacob O. Wobbrock. 2010. Automatically generating personalized user interfaces with SUPPLE. *Journal of Artificial Intelligence* 174, 12–13: 910–950. https://doi.org/10.1016/j.artint.2010.05.005
25. Krzysztof Z. Gajos, Jacob O. Wobbrock, and Daniel S. Weld. 2008. Improving the performance of motor-impaired users with automatically-generated, ability-based interfaces. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '08), 1257–1266. https://doi.org/10.1145/1357054.1357250
26. Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition* (CVPR '14), 580–587. https://doi.org/10.1109/CVPR.2014.81
27. Mayank Goel, Leah Findlater, and Jacob Wobbrock. 2012. WalkType: Using accelerometer data to accomodate situational impairments in mobile touch screen text entry. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '12), 2687–2696. https://doi.org/10.1145/2207676.2208662
28. Tiago João Vieira Guerreiro, Hugo Nicolau, Joaquim Jorge, and Daniel Gonçalves. 2010. Assessing mobile touch interfaces for tetraplegics. In *Proceedings of the ACM Conference on Human*

Computer Interaction with Mobile Devices and Services (MobileHCI '10), 31–34. https://doi.org/10.1145/1851600.1851608
29. Tiago Guerreiro, Hugo Nicolau, Joaquim Jorge, and Daniel Gonçalves. 2010. Towards accessible touch interfaces. In *Proceedings of the ACM SIGACCESS Conference on Computers and Accessibility* (ASSETS '10), 19–26. https://doi.org/10.1145/1878803.1878809
30. Foad Hamidi, Kellie Poneres, Aaron Massey, and Amy Hurst. 2018. Who Should Have Access to My Pointing Data?: Privacy Tradeoffs of Adaptive Assistive Technologies. In *Proceedings of the 20th International ACM SIGACCESS Conference on Computers and Accessibility* (ASSETS '18), 203–216. https://doi.org/10.1145/3234695.3239331
31. J. A. Hartigan and M. A. Wong. 1979. Algorithm AS 136: A K-Means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 28, 1: 100–108. https://doi.org/10.2307/2346830
32. Niels Henze, Enrico Rukzio, and Susanne Boll. 2011. 100,000,000 Taps: Analysis and improvement of touch performance in the large. In *Proceedings of the ACM Conference on Human Computer Interaction with Mobile Devices and Services* (MobileHCI '11), 133–142. https://doi.org/10.1145/2037373.2037395
33. James J. Higgins and S Tashtoush. 1994. An aligned rank transform test for interaction. *Nonlinear World* 1, 2: 201–211.
34. Sture Holm. 1979. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics* 6, 2: 65–70.
35. Christian Holz and Patrick Baudisch. 2010. The generalized perceived input point model and how to double touch accuracy by extracting fingerprints. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '10), 581–590. https://doi.org/10.1145/1753326.1753413
36. Christian Holz and Patrick Baudisch. 2011. Understanding touch. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '11), 2501–2510. https://doi.org/10.1145/1978942.1979308
37. Curt B. Irwin and Mary E. Sesto. 2012. Performance and touch characteristics of disabled and non-disabled participants during a reciprocal tapping task using touch screen technology. *Applied Ergonomics* 43, 6: 1038–1043. https://doi.org/10.1016/j.apergo.2012.03.003
38. Shaun K. Kane, Chandrika Jayant, Jacob O. Wobbrock, and Richard E. Ladner. 2009. Freedom to roam: A study of mobile device adoption and accessibility for people with visual and motor disabilities. In *Proceedings of the ACM SIGACCESS Conference on Computers and Accessibility* (Assets '09), 115–122. https://doi.org/10.1145/1639642.1639663
39. Shaun K. Kane, Jacob O. Wobbrock, and Richard E. Ladner. 2011. Usable gestures for blind people: Understanding preference and performance. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '11), 413–422. https://doi.org/10.1145/1978942.1979001
40. Shaun K. Kane, Jacob O. Wobbrock, and Ian E. Smith. 2008. Getting off the treadmill: Evaluating walking user interfaces for mobile devices in public spaces. In *Proceedings of the ACM Conference on Human Computer Interaction with Mobile Devices and Services* (MobileHCI '08), 109–118. https://doi.org/10.1145/1409240.1409253
41. Simeon Keates and P. John Clarkson. 2003. Countering design exclusion through inclusive design. In *Proceedings of the ACM Conference on Universal Usability* (CUU '03), 69–76. https://doi.org/10.1145/957205.957218
42. Simeon Keates, P. John Clarkson, Lee-Anne Harrison, and Peter Robinson. 2000. Towards a practical inclusive design approach. In *Proceedings on the ACM Conference on Universal Usability* (CUU '00), 45–52. https://doi.org/10.1145/355460.355471
43. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. In *Proceedings of the International Conference on Neural Information*

Processing Systems (NIPS'12), 1097–1105. Retrieved May 26, 2018 from http://dl.acm.org/citation.cfm?id=2999134.2999257
44. S. Lawrence, C. L. Giles, Ah Chung Tsoi, and A. D. Back. 1997. Face recognition: a convolutional neural-network approach. *IEEE Transactions on Neural Networks* 8, 1: 98–113. https://doi.org/10.1109/72.554195
45. Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86, 11: 2278–2324. https://doi.org/10.1109/5.726791
46. Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553: 436–444. https://doi.org/10.1038/nature14539
47. Min Lin, Rich Goldman, Kathleen J. Price, Andrew Sears, and Julie Jacko. 2007. How do people tap when walking? An empirical investigation of nomadic data entry. *International Journal of Human-Computer Studies* 65, 9: 759–769. https://doi.org/10.1016/j.ijhcs.2007.04.001
48. R. C. Littell, P. R. Henry, and C. B. Ammerman. 1998. Statistical analysis of repeated measures data using SAS procedures. *Journal of Animal Science* 76, 4: 1216–1231.
49. J. Long, E. Shelhamer, and T. Darrell. 2015. Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3431–3440. https://doi.org/10.1109/CVPR.2015.7298965
50. Ronald L. Mace, Graeme J. Hardie, and Jaine P. Place. 1991. Accessible environments: Towards universal design. In *Design Intervention: Toward a More Humane Architecture*. Van Nostrand Reinhold, New York, NY, USA, 155–176.
51. Meethu Malu, Pramod Chundury, and Leah Findlater. 2018. Exploring accessible smartwatch interactions for people with upper body motor impairments. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (CHI '18), 488:1–488:12. https://doi.org/10.1145/3173574.3174062
52. Tom M. Mitchell. 1997. *Machine Learning*. McGraw-Hill Education, New York.
53. Karyn Moffatt and Joanna McGrenere. 2010. Steadied-bubbles: Combining techniques to address pen-based pointing errors for younger and older adults. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '10), 1125–1134. https://doi.org/10.1145/1753326.1753495
54. Kyle Montague, Vicki L. Hanson, and Andy Cobley. 2012. Designing for individuals: Usable touch-screen interaction through shared user models. In *Proceedings of the ACM SIGACCESS Conference on Computers and Accessibility* (ASSETS '12), 151–158. https://doi.org/10.1145/2384916.2384943
55. Kyle Montague, Hugo Nicolau, and Vicki L. Hanson. 2014. Motor-impaired touchscreen interactions in the wild. In *Proceedings of the ACM SIGACCESS Conference on Computers and Accessibility* (ASSETS '14), 123–130. https://doi.org/10.1145/2661334.2661362
56. Martez E. Mott, Jane E., Cynthia L. Bennett, Edward Cutrell, and Meredith Ringel Morris. 2018. Understanding the Accessibility of smartphone photography for people with motor impairments. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '18), 520:1–520:12. https://doi.org/10.1145/3173574.3174094
57. Martez E. Mott, Radu-Daniel Vatavu, Shaun K. Kane, and Jacob O. Wobbrock. 2016. Smart Touch: Improving touch accuracy for people with motor Impairments with template matching. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '16), 1934–1946. https://doi.org/10.1145/2858036.2858390
58. Josip Musić, Daryl Weir, Roderick Murray-Smith, and Simon Rogers. 2016. Modelling and correcting for the impact of the gait cycle on touch screen typing accuracy. *mUX: The Journal of Mobile User Experience* 5, 1: 1. https://doi.org/10.1186/s13678-016-0002-3

59. Maia Naftali and Leah Findlater. 2014. Accessibility in context: Understanding the truly mobile experience of smartphone users with motor impairments. In *Proceedings of the ACM SIGACCESS Conference on Computers and Accessibility* (ASSETS '14), 209–216. https://doi.org/10.1145/2661334.2661372
60. Alexander Ng. 2014. The effects of encumbrance on mobile interactions. In *Proceedings of the ACM Conference on Human-computer Interaction with Mobile Devices & Services* (MobileHCI '14), 405–406. https://doi.org/10.1145/2628363.2634268
61. Alexander Ng, John Williamson, and Stephen Brewster. 2015. The effects of encumbrance and mobility on touch-based gesture interactions for mobile phones. In *Proceedings of the ACM Conference on Human-computer Interaction with Mobile Devices and Services* (MobileHCI '15), 536–546. https://doi.org/10.1145/2785830.2785853
62. Pekka Parhi, Amy K. Karlson, and Benjamin B. Bederson. 2006. Target size study for one-handed thumb use on small touchscreen devices. In *Proceedings of the ACM Conference on Human-computer Interaction with Mobile Devices and Services* (MobileHCI '06), 203–210. https://doi.org/10.1145/1152215.1152260
63. R. L. Potter, L. J. Weldon, and B. Shneiderman. 1988. Improving the accuracy of touch screens: An experimental evaluation of three strategies. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '88), 27–32. https://doi.org/10.1145/57167.57171
64. K. C. Salter and R. F. Fawcett. 1993. The art test of interaction: a robust and powerful rank test of interaction in factorial models. *Communications in Statistics - Simulation and Computation* 22, 1: 137–153. https://doi.org/10.1080/03610919308813085
65. Thiago Santini, Wolfgang Fuhl, and Enkelejda Kasneci. 2017. CalibMe: Fast and Unsupervised Eye Tracker Calibration for Gaze-Based Pervasive Human-Computer Interaction. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (CHI '17), 2594–2605. https://doi.org/10.1145/3025453.3025950
66. Julia Schwarz, Scott Hudson, Jennifer Mankoff, and Andrew D. Wilson. 2010. A framework for robust and flexible handling of inputs with uncertainty. In *Proceedings of the ACM Symposium on User Interface Software and Technology* (UIST '10), 47–56. https://doi.org/10.1145/1866029.1866039
67. Julia Schwarz, Jennifer Mankoff, and Scott Hudson. 2011. Monte Carlo methods for managing interactive state, action and feedback under uncertainty. In *Proceedings of the ACM Symposium on User Interface Software and Technology* (UIST '11), 235–244. https://doi.org/10.1145/2047196.2047227
68. Julia Schwarz, Jennifer Mankoff, and Scott E. Hudson. 2015. An architecture for generating interactive feedback in probabilistic user interfaces. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '15), 2545–2554. https://doi.org/10.1145/2702123.2702228
69. Andrew Sears, Min Lin, Julie Jacko, and Yan Xiao. 2003. When computers fade … pervasive computing and situationally-induced impairments and disabilities. 1298–1302.
70. Andrew Sears and Ben Shneiderman. 1991. High precision touchscreens: Design strategies and comparisons with a mouse. *International Journal of Man-Machine Studies* 34, 4: 593–613. https://doi.org/10.1016/0020-7373(91)90037-8
71. Andrew Sears and Mark Young. 2003. Physical disabilities and computing technologies: an analysis of impairments. In Julie A. Jacko and Andrew Sears (eds.). L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 482–503. Retrieved September 20, 2018 from http://dl.acm.org/citation.cfm?id=772072.772105
72. Mary E. Sesto, Curtis B. Irwin, Karen B. Chen, Amrish O. Chourasia, and Douglas A. Wiegmann. 2012. Effect of touch screen button size and spacing on touch characteristics of users with and without disabilities. *Human Factors* 54, 3: 425–436.
73. B. Shneiderman. 1991. Touch screens now offer compelling uses. *IEEE Software* 8, 2: 93–94. https://doi.org/10.1109/52.73754

74. Shari Trewin, Cal Swart, and Donna Pettick. 2013. Physical accessibility of touchscreen smartphones. In *Proceedings of the ACM SIGACCESS Conference on Computers and Accessibility* (ASSETS '13), 19:1–19:8. https://doi.org/10.1145/2513383.2513446
75. Radu-Daniel Vatavu, Lisa Anthony, and Jacob O. Wobbrock. 2012. Gestures as point clouds: A $P recognizer for user interface prototypes. In *Proceedings of the ACM International Conference on Multimodal Interaction* (ICMI '12), 273–280. https://doi.org/10.1145/2388676.2388732
76. Daniel Vogel and Ravin Balakrishnan. 2010. Occlusion-aware interfaces. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '10), 263–272. https://doi.org/10.1145/1753326.1753365
77. Daniel Vogel and Patrick Baudisch. 2007. Shift: A technique for operating pen-based interfaces using touch. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '07), 657–666. https://doi.org/10.1145/1240624.1240727
78. Chat Wacharamanotham, Jan Hurtmanns, Alexander Mertens, Martin Kronenbuerger, Christopher Schlick, and Jan Borchers. 2011. Evaluating swabbing: A touchscreen input method for elderly users with tremor. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '11), 623–626. https://doi.org/10.1145/1978942.1979031
79. Daryl Weir. 2012. Machine learning models for uncertain interaction. In *Adjunct Proceedings of the ACM Symposium on User Interface Software and Technology* (UIST Adjunct Proceedings '12), 31–34. https://doi.org/10.1145/2380296.2380313
80. Daryl Weir, Simon Rogers, Roderick Murray-Smith, and Markus Löchtefeld. 2012. A user-specific machine learning approach for improving touch accuracy on mobile devices. In *Proceedings of the ACM Symposium on User Interface Software and Technology* (UIST '12), 465–476. https://doi.org/10.1145/2380116.2380175
81. Daniel Wigdor, Clifton Forlines, Patrick Baudisch, John Barnwell, and Chia Shen. 2007. Lucid Touch: A see-through mobile device. In *Proceedings of the ACM Symposium on User Interface Software and Technology* (UIST '07), 269–278. https://doi.org/10.1145/1294211.1294259
82. Jacob O. Wobbrock. 2014. Improving pointing in graphical user interfaces for people with motor impairments through ability-based design. In *Assitive Technologies and Computer Access for Motor Disabilities*. IGI Global, Hershey, PA, 206–253.
83. Jacob O. Wobbrock, Htet Htet Aung, Brandon Rothrock, and Brad A. Myers. 2005. Maximizing the guessability of symbolic input. In *Extended Abstracts of the ACM Conference on Human Factors in Computing Systems* (CHI EA '05), 1869–1872. https://doi.org/10.1145/1056808.1057043
84. Jacob O. Wobbrock, Leah Findlater, Darren Gergle, and James J. Higgins. 2011. The Aligned Rank Transform for nonparametric factorial analyses using only ANOVA procedures. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '11), 143–146. https://doi.org/10.1145/1978942.1978963
85. Jacob O. Wobbrock, James Fogarty, Shih-Yen (Sean) Liu, Shunichi Kimuro, and Susumu Harada. 2009. The Angle Mouse: Target-agnostic dynamic gain adjustment based on angular deviation. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '09), 1401–1410. https://doi.org/10.1145/1518701.1518912
86. Jacob O. Wobbrock, Krzysztof Z. Gajos, Shaun K. Kane, and Gregg C. Vanderheiden. 2018. Ability-based design. *Communications of the ACM* 61, 6: 62–71. https://doi.org/10.1145/3148051
87. Jacob O. Wobbrock, Shaun K. Kane, Krzysztof Z. Gajos, Susumu Harada, and Jon Froehlich. 2011. Ability-based design: Concept, principles and examples. *ACM Transactions on Accessible Computing* 3, 3: 9:1–9:27. https://doi.org/10.1145/1952383.1952384
88. Jacob O. Wobbrock, Meredith Ringel Morris, and Andrew D. Wilson. 2009. User-defined gestures for surface computing. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '09), 1083–1092. https://doi.org/10.1145/1518701.1518866

89. Jacob O. Wobbrock, Brad A. Myers, and John A. Kembel. 2003. EdgeWrite: A stylus-based text entry method designed for high accuracy and stability of motion. In *Proceedings of the ACM Symposium on User Interface Software and Technology* (UIST '03), 61–70. https://doi.org/10.1145/964696.964703
90. Jacob O. Wobbrock, Andrew D. Wilson, and Yang Li. 2007. Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes. In *Proceedings of the ACM Symposium on User Interface Software and Technology* (UIST '07), 159–168. https://doi.org/10.1145/1294211.1294238
91. Yu Zhong, Astrid Weber, Casey Burkhardt, Phil Weaver, and Jefferey P. Bigham. 2015. Enhancing Android accessibility for users with hand tremor by reducing fine pointing and steady tapping. In *Proceedings of the Web for All Conference* (W4A '15), 1–10.

Appendix A

$P Point-Cloud Recognizer Extended to Different Point Cardinalities

In the following pseudocode, I list a new variant of the $P point-cloud gesture recognizer [75] adapted for the problem of predicting touch input locations from varied touch behavior. The new $P variant can match an unequal number of touch areas (i.e., ellipses) that make up templates and candidates.

GREEDY-CLOUD-MATCH-UNEQUAL (POINTS *candidate*, POINTS *template*)

```
1.  n ← MAX(candidate.length, template.length)
2.  ε ← .50
3.  step ←floor(n^(1−ε))
4.  min ← ∞
5.  ptsLeft ← n
6.  if candidate.length > template.length then
7.      for i = 0 to n step step do
8.          d ← CLOUD-DISTANCE-UNEQUAL(template,
            candidate, i, ptsLeft)
9.          min ← MIN(min, d)
10. else
11.     for i = 0 to n step step do
12.         d ← CLOUD-DISTANCE-
            UNEQUAL(candidate, template, i, ptsLeft)
13.         min ← MIN (min, d)
14. return min
```

DETERMINE-REMAINING (bool[ ] *matched*, POINTS *points,* int *diff*)

```
1.  remaining ← new POINTS[diff]
2.  index ← 0
3.  for each i such that not matched[i] do
4.      remaining[index] ← points[i]
5.      index ← index + 1
6.  return remaining
```

CLOUD-DISTANCE-UNEQUAL (POINTS *matcher*, POINTS *matchee*, int *start*, int *ptsLeft*)

```
1.  if ptsLeft == 0 then
2.      return 0
3.  else
4.      m ← matcher.length
5.      n ← matchee.length
6.      matched ← new bool[n]
7.      sum ← 0
8.      i ← start
9.      do
10.         min ← ∞
11.         index ← -1
12.         for each j such that not matched[j] do
13.             d ← EUCLIDEAN-DISTANCE(matcher_i,
                matchee_j)
14.             if d < min then
15.                 min ← d
16.                 index ← j
17.         matched[index] ← true
18.         ptsLeft ← ptsLeft - 1
19.         weight ← 1 – ((i – start + m) MOD m) /
            m
20.         sum ← sum + weight × min
21.         i ← (i + 1) MOD m
22.     until i == start
23.     diff ← n – m
24.     remaining ← DETERMINE-REMAINING(matched,
        matchee, diff)
25.     if diff < m then
26.         return sum + CLOUD-DISTANCE-
            UNEQUAL(remaining, matcher, 0,
            ptsLeft)
27.     else
28.         return sum + CLOUD-DISTANCE-
            UNEQUAL(matcher, remaining, 0,
            ptsLeft)
```