RESEARCH-ARTICLE

Smart Touch: Improving Touch Accuracy for People with Motor Impairments with Template Matching

MARTEZ E. MOTT, University of Washington, Seattle, WA, United States

RADU-DANIEL VATAVU, Stefan cel Mare University of Suceava, Suceava, Romania

SHAUN K. KANE, University of Colorado Boulder, Boulder, CO, United States

JACOB O. WOBBROCK, University of Washington, Seattle, WA, United States

Published: 07 May 2016

CHI'16: CHI Conference on Human Factors in Computing Systems
May 7 - 12, 2016
California, San Jose, USA

Conference Sponsors:
SIGCHI

Smart Touch: Improving Touch Accuracy for People with Motor Impairments with Template Matching

Martez E. Mott[1], Radu-Daniel Vatavu[2], Shaun K. Kane[3] and Jacob O. Wobbrock[1]

[1]The Information School
DUB Group
University of Washington
Seattle, WA 98195 USA
{memott, wobbrock}@uw.edu

[2]MintViz Lab | MANSiD
Research Center
University Stefan cel
Mare of Suceava
Suceava 720229, Romania
vatavu@eed.usv.ro

[3]University of Colorado, Boulder
Boulder, CO 80309 USA
shaun.kane@colorado.edu

ABSTRACT

We present two contributions for improving the accessibility of touch screens for people with motor impairments. First, we provide an exploration of the touch behaviors of 10 people with motor impairments, *e.g.*, we describe how touching with the back or sides of the hand, with multiple fingers, or with the knuckles creates varied multi-point touches. Second, we introduce *Smart Touch*, a novel template-matching technique for touch input that maps any number of arbitrary contact-areas to a user's intended (x, y) target location. The result is that users with motor impairments can touch however their abilities allow, and Smart Touch will resolve their intended touch point. Smart Touch therefore allows users to touch targets in whichever ways are most comfortable and natural for them. In an experiment, we found that Smart Touch predicted the (x, y) coordinates of users' intended target locations over three times closer to actual intended targets than the native Land-on and Lift-off techniques reported by the built-in touch sensors found in the Microsoft PixelSense interactive tabletop. This result is an important step toward improving touch accuracy for people with motor impairments and others for whom touch screen operation was previously difficult or impossible.

Author Keywords

Ability-based design; accessibility; motor impairment; $P recognizer; target acquisition; touch input; touch screens.

ACM Classification Keywords

H.5.2. [Information interfaces and presentation]: User interfaces – *input devices and strategies*. K.4.2. [Computers and society]: Social issues – *assistive technologies for persons with disabilities*.

CHI'16, May 07-12, 2016, San Jose, CA, USA
 ISBN 978-1-4503-3362-7/16/05…$15.00
DOI: http://dx.doi.org/10.1145/2858036.2858390

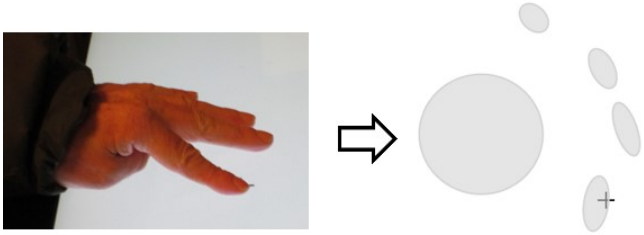

Figure 1. Touching with multiple fingers or various parts of the hand (left) creates various contact regions (right). Current touch screens are not designed to accommodate this kind of touch input when the user's goal is to activate just a single (x,y) point.

INTRODUCTION

Touch is one of the most dominant ways users interact with modern computing devices. The predominance of touch can be attributed to the pervasiveness of touch-enabled devices, such as smartphones, tablets, interactive tables, interactive wall displays, and public kiosks. The simple and direct nature of touch also makes it a popular form of interaction.

Although touch has been widely adopted, it remains largely inaccessible for many people with motor impairments [2,5,8,16,17,21,27,35]. The requirement that a user be able to suspend an arm, extend a finger, and land and lift accurately on a touch-sensitive surface is beyond the abilities of many people with motor impairments. Instead, many people with motor impairments touch inadvertently with multiple parts of their hand, or are able to use only the back of their hand or even their elbow. Our perspective is that the touch screen access barrier does not inherently lie with users, but with the implicit ability-assumptions embedded in the design of touch screens. It is because the ability-assumptions do not match the actual abilities of many people with motor impairments that touch screens remain inaccessible. Adopting a perspective of *ability-based design* [42,44], we seek to upend these assumptions by making touch screens more amenable to a much wider range of users' touch-abilities, enabling more users to benefit from this technology.

Previous research has investigated the accessibility of touch screens for people with motor impairments in both lab [8,10,16,17,21,35] and field settings [2,22,26,27]. However, little is known about the touch behavior of people who use various parts of their hand, not just a single finger, when interacting with a touch screen (Figure 1, left.) There has also

been little advancement into new techniques for improving touch performance for users with motor impairments, especially techniques designed to accommodate varied multi-touch input (see Figure 1, right.)

To further our understanding of touch input and to improve touch accuracy for people with motor impairments, we present in this work: (1) an exploration of the touch behavior of 10 people with motor impairments, and (2) a novel template-matching algorithm for touch input called *Smart Touch*. Specifically, we identify and describe two touch behaviors that challenge the assumptions of current touch screens and, ultimately, impact their accessibility. First, utilizing various parts of the hand creates a multi-contact problem, where multiple contact areas are registered but only a single point is intended by the user. Second, the difficulty of accurately landing and lifting is exhibited by users who do not possess the motor control to land and lift within their intended target. Instead, some users go through an extended *touch process*, where the screen is used as a stabilizing surface against which some users slide, resulting in multiple touches over time as users approach their target.

To improve upon the limitations of touch screens, we created Smart Touch, a novel three-step user-specific template-matching algorithm that maps any number of arbitrary touch-areas to a user's intended (x, y) target location. First, Smart Touch analyzes a user's touch process to extract the most relevant touch data. Second, a template matcher is employed to match the extracted touch data to previously observed touch instances that were captured as training examples. Third, the location of the user's intended (x, y) target is predicted by adding an offset to the weighted centroid of the extracted touch data based on the best-matched template.

We evaluated the effectiveness of Smart Touch using touch data collected from 10 participants with motor impairments. Our results showed that Smart Touch predicted (x, y) coordinates of the users' intended target locations over three times closer to the intended target than the native Land-on and Lift-off techniques reported by the built-in touch sensors found in the Microsoft PixelSense interactive tabletop. This increase in touch accuracy brings us closer to accessible and operable touch screens for people with motor impairments.

The contributions of this work are: (1) an empirical characterization of the touch behaviors of 10 people with motor impairments; (2) a novel template-matching algorithm called Smart Touch, including a general extension of the $P point-cloud recognizer [36] to point-clouds with different point cardinalities; and (3) empirical results from a study of Smart Touch comparing it to the Microsoft PixelSense using the touch data collected from our 10 participants. This work takes a significant step toward realizing the vision of ability-based design [42,44] for touch screens and for people with motor impairments.

RELATED WORK

This work is motivated by prior research on accessible touch screen technology, high precision touch techniques, theories of touch input, probabilistic input, and ability-based design.

Understanding Touch Screen Accessibility for People with Motor Impairments

There have been numerous investigations into understanding the accessibility of touch screens for people with motor impairments. Research has provided more understanding into the everyday use of touch screens by people with motor impairments [2,22,26,27]. In these investigations, touch-enabled mobile devices have been acknowledged as improving independence and creating a sense of empowerment [2,22]. Findings also describe the accessibility challenges of current touch screens. Notably, in their analysis of YouTube videos with people physical disabilities interacting with touch screens, Anthony *et al*. [2] observed that individuals encountered difficulties using standard touch behaviors, and instead adopted a variety of different interaction styles, including the use of multiple fingers, hands, fists, and knuckles.

In addition to touch screen use "in the wild," researchers have also examined touch screen use in the lab [8,10,13,16,17,21,35,45]. Guerreiro *et al.* [16,17] explored how different interaction techniques and target properties impact the ability of users to acquire targets. Trewin *et al*. [35] found that for people with motor impairments, performing sliding and tapping gestures resulted in more errors and accidental activation of other touch screen features, such as zooming. They also found that users would slide their finger along the screen for stability as they approached their target. Our research builds upon and extends this prior research by providing an analysis of non-single-finger touch, a form of interaction reported in previous research [2,35] but not thoroughly investigated.

The research to-date in understanding touch accessibility has been both descriptive (detailing the difficulties of performing certain touch screen interactions) and prescriptive (providing design guidelines such as minimum widget size). Little advancement, however, has been made in the way of new algorithms, techniques, and approaches for fundamentally improving the accuracy of touch input for people with motor impairments. Notable exceptions include Biswas and Langdon [5], who created an algorithm to improve the pointing performance of users with motor impairments by measuring hand strength. Montague *et al*. [26] introduced a novel tap gesture recognizer to improve the accuracy of tap recognition on touch-enabled devices. In another invention, Montague *et al*. introduced the Shared User Modeling Framework [25], an adaptive framework that aimed to improve touch accessibility across devices and applications. Wacharamanotham *et al*. evaluated *Swabbing* [38], a technique for elderly adults to acquire targets by dragging their finger across a target rather than discretely tapping it. Unlike these techniques, which focused primarily on single finger touch input, Smart Touch improves touch accuracy while allowing users to interact with the screen *in whichever way is most comfortable and natural for them*.

High-precision Touch Screen Interaction Techniques

Accurately acquiring targets using a finger has been a limitation of touch screens since their inception, thanks in large part to the "fat finger" occlusion and precision problems [37]. As a result, researchers have developed several techniques to help users of any abilities to acquire targets. For instance, Potter *et al*. [29] investigated the *offset-cursor* with three different selection techniques, *land-on*, *first-contact*, and *take-off*. Sears and Shneiderman [34] added stabilization to the offset-cursor to improve pointing performance. Albinsson and Zhai [1] created *Cross Keys* and *Precision-Handle*, two pointing techniques that allow for pixel-level precision pointing. Benko *et al.* developed *Dual Finger Selections* [4], a collection of five techniques which leverage multi-touch by allowing the use of two fingers to acquire small targets. *Shift* by Vogel and Baudisch [37] is a technique that creates a callout when a finger occludes small targets, allowing such targets to be acquired quickly and accurately. Wigdor *et al.* invented *LucidTouch* [41], a device that solves the occlusion problem by allowing users to interact with targets by touching the back of the device.

All of these techniques require users to possess fine motor control skills, making them inaccessible to many people with motor impairments. In contrast, Smart Touch was designed specifically to improve touch accuracy for users with reduced and poor motor control.

Probabilistic Input

Researchers have exploited the uncertain nature of touch input to improve its accuracy. Schwarz *et al*. proposed methods [31], frameworks [30], and an architecture [32] to handle uncertain input techniques, including touch input. Bi and Zhai developed *Bayesian Touch* [7], a statistical target acquisition technique based on Bayes' rule and the bivariate Gaussian distribution principle of finger touch [6]. Weir *et al*. [39,40] created a machine learning approach that improved touch accuracy using the device's reported (x, y) touch location and the raw sensor data.

Although probabilistic methods have been shown to improve touch accuracy, they also require the system to be *target-aware* [3], requiring the system to have knowledge of all on-screen targets. Target-aware systems are difficult to implement in the real-world because of the engineering and theoretical challenge they pose [9]. Smart Touch, however, is *target-agnostic* [43], requiring no information of the locations and dimensions of on-screen targets. As a result, Smart Touch has the potential to be much more easily deployed on current touch-enabled systems.

Utilizing Users' Mental Model of Touch

Work by Holz and Baudisch [19,20] showed that touch screens introduce a systematic error by not taking into account users' mental model of touch when determining their intended touch point. The authors were able to improve touch accuracy by employing a model based on the visual features of users' fingers. Their approach demonstrated that using non-disabled users' mental models of touch could improve touch accuracy. However, little is known about the mental models of touch for people with motor impairments. Our analysis provides a detailed view into the touch behaviors of people with motor impairments, while Smart Touch utilizes this knowledge to improve the accuracy and operability of touch screens.

Ability-Based Design

Our work is motivated by the concepts and principles of *ability-based design* [42,44], an approach to achieving accessible design that emphasizes discovering what users *can* do, instead of focusing on what they cannot do, and then creating systems that can adapt or be adapted to supporting the actual abilities of users. A key tenet of ability-based design is that the burden of adaptation should be placed on the system rather than on the user. Perhaps the best example of ability-based design is *SUPPLE* [14,15], a system that automatically generates user interfaces to best accommodate the unique mouse-pointing abilities of users.

Smart Touch embodies the concepts and principles of ability-based design by allowing users to touch "as they are" with whatever part of their hand they prefer and in the manner which is most comfortable and natural for them. Touch screens and their algorithms come to bear the burden of enabling a much wider range of touch-abilities to be effective for their operation than ever before.

EXPLORATION OF MOTOR-IMPAIRED TOUCH

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

Participants

We recruited 10 people (4 female, 6 male, average age 52.5 years, *SD*=8.46) with motor impairments from a local organization that provides rehabilitation, job placement, and community living for people with physical disabilities. Six participants reported having greater control in their right arm and hand and four reported greater control in their left arm and hand. All participants were paid $30 for their participation, which lasted about one hour. Additional details about our participants can be found in Table 1.

Apparatus

Touch data was collected in an experiment testbed developed in C# .NET 4.5. All sessions were conducted on a Microsoft PixelSense interactive table running Windows 7. The testbed captured and logged all touch events registered by the sensor in the PixelSense. We selected the PixelSense as our platform because of the wealth of information provided by the touch sensor. Each registered touch is presented as an ellipse with a major and minor axis, and an orientation. To accommodate participants in wheelchairs, the PixelSense was placed on an adjustable-height table. The table was adjusted before every session to a height most comfortable for each participant.

ID	Age	Sex	Touch method	Health condition	Self-reported impairments†										
					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 our participants. Categories used for self-reporting impairments were from Findlater *et al.* [12].

Figure 2. A participant performs tasks on the Microsoft PixelSense.

Procedure

Seven participants engaged in study sessions in a computer lab at their organization and three engaged in sessions in our university lab. Each session lasted approximately one hour. Sessions began with the experimenter explaining the capabilities of the PixelSense and what information would be collected during the session. Next, the experimenter asked the participant to demonstrate where he or she was most comfortable interacting on the screen. The experimenter then drew a rectangular region on the screen around the area the participant indicated was comfortable to reach. The size and placement of the region varied by participant. The rectangular region acted as the designated interactive space and targets only appeared inside that space. A crosshairs was then drawn by the testbed inside the interactive space.

The participant was instructed to touch the center of the crosshairs in whichever way was most comfortable and natural for them. The experimenter asked the participant to demonstrate touching the crosshairs five times for practice. After, the data collection trials began.

In each trial, a crosshairs was placed randomly inside the interactive region. A trial began when the first touch event was registered and ended after all registered touches had been removed and no new touches were detected for one second. This one second timeout was used to prevent trials from accidently advancing when registered touches would suddenly disappear and reappear due to the touch behavior of our participants. There was a three second countdown between trials.

The participant was asked to complete trials at a pace that was most comfortable for him or her. Based on pilot testing, we set the maximum number of trials to 110. Due to fatigue, however, many of our participants could not complete all 110 trials. Participants were instructed to complete as many trials as they could. On average, participants completed 94.4 trials.

Results

We collected a total of 944 trials from our 10 participants. Eleven trials were discarded because of a sensor error in reporting participants' lift-off locations. In total, 932 trials were analyzed; see Table 2 for the number of trials completed by each participant. In the following sections, we provide an analysis of the collected touch data, detailing how various aspects of our participants' touch behavior affects their ability to interact with touch screens.

Concurrent Touches

Across all participants, the average number of concurrent touches (*i.e.*, the number of simultaneous touches registered by the sensor at any given time) was 2.30 (*SD*=1.38). To calculate the average number of concurrent touches, we portioned each trial into frames (a definition of a "frame" is provided in the next section) and we counted the number of active touches in each frame.

Figure 3. P6 interacting with the PixelSense. P6 is using the entire left edge of his hand to interact with the touch screen.

Many of our participants averaged more than one concurrent touch during a given trial. The presence of multiple concurrent touches is a result of the use of multiple fingers (see Figure 1) and various parts of the hand (Figure 3) when interacting with the screen. Distributions of the average number of concurrent touches for each participant are shown in Figure 4.

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.

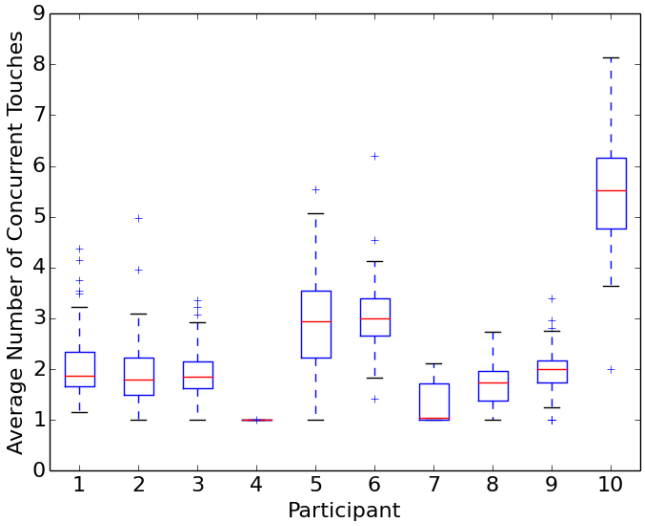

Figure 4. Distribution of the average number of concurrent touches per trial. Due to their touch behavior, participants impacted the screen with various parts of the hand, resulting in multiple registered touches.

Trial Duration

The average duration of a trial across our participants was 1870.82 ms (*SD*=3029.16). A trial began when the first touch event was registered and ended when all registered touches were removed. (The one second timeout used to ensure trials did not advance prematurely was not included in the trial duration.) Our results show that average target acquisition times for many of our participants are much longer for motor-impaired users than for what the literature reports for non-disabled users (600 ms to 1200 ms, depending on target size [28]). Mean trial duration times per participant are shown in Table 2.

Touch-Down and Touch-Up

In current touch screen systems, a target is acquired when a user successfully lands and lifts within the target bounds. To see how accurately our participants could land and lift near the displayed crosshairs in each trial, we measured the distance from the center of the ellipses that represent the first and last registered touches to the center of each crosshairs. The mean touch-down distance for our participants was 9.71 cm (*SD*=8.69), and the mean touch-up distance was 9.97 cm (*SD*=10.25). On the Microsoft PixelSense, a centimeter corresponded to about 22 pixels. Mean distances

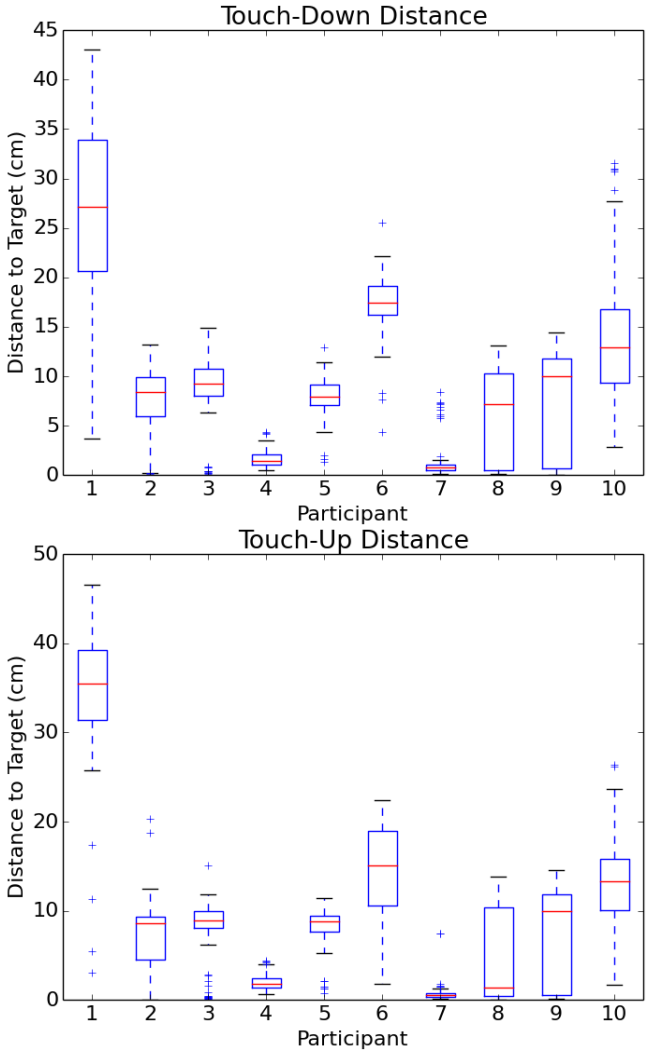

Figure 5. Distribution of distances between the touch-down (top) and touch-up (bottom) locations to the target center. Note that the distances are measured in centimeters. A centimeter on the Microsoft PixelSense is equivalent to about 22 pixels.

for touch-up and touch-down are shown in Table 2. Figure 5 shows the distribution of our participants' touch-down and touch-up distances.

Discussion

Our data provides some important insights into the touch behavior of users with motor impairments. With respect to

touch-down and touch-up events, our participants' average touch-down and touch-up distances from the target center were much higher than those reported in previous studies for non-disabled users; for instance, the average error offset for the non-disabled participants in Holz and Baudisch's study was only 4 mm [20], compared to 97+ mm in our study. The much higher touch distances exhibited by our participants is a result of two observed behaviors. The first is that participants accidently impacted the screen with their palm or other fingers before they were prepared to touch the crosshairs. The same is true as they lifted their fingers or hand off the screen. Prematurely impacting the screen is a behavior also observed in other studies exploring touch screen use by people with motor impairments [35]. The second behavior is that participants would use the screen for support by sliding on it as they approached their target. Participants who used this approach experienced difficulties moving their hand and arm freely through the air. Participants would also use the screen for stability during lift-off, as they would move their hand away from the target and lift once it was comfortable for them. As a result of exhibiting these two behaviors, the touch-down and touch-up locations registered by the system were quite far from the intended target. The landing and lifting distances were well beyond the widths of current touch screen widgets. For example, a typical 50 pixel button on the Microsoft PixelSense measures only 2.31 cm wide, a mere 23.79% of our average distance from target. Even previously recommended target sizes for motor-impaired users would not be large enough to accommodate the touch behaviors of our participants; for example, see Guerreiro *et al*. [17], who recommended a minimum target size of 12 mm.

Another behavior exhibited by our participants was dragging multiple fingers and various parts of the hand while interacting with the screen. Some participants tried to use a single finger, but lacked the dexterity to touch with a fully extended finger. Instead, multiple fingers and parts of the hand impacted the screen throughout the trial, resulting in multiple concurrent touches. Other participants were unable to extend and fully control *any* of their fingers, using the fist or hand to interact with the screen. This behavior also resulted in multiple concurrent touches, as the system attempted to fit contact points to the various parts of the hand touching the screen (Figure 6). Current touch screen systems are not designed to accommodate varied multi-touch input for target acquisition tasks and previous research has found that it is common for users with motor impairments to accidently trigger multi-touch gestures when inadvertently touching with more than one finger [35].

To improve touch screen accuracy, we must view the touch behavior of people with motor impairments holistically. Current touch screens place too much emphasis on the first and last contact points a user makes on the screen. Current touch screens also assume users can touch the screen with a fully extended finger, resulting in one contact point the system can use to infer which target the user is trying to acquire. Instead of focusing on the land-on and lift-off locations of a single contact point, systems need to take into account the user's entire *touch process*. A user's touch process includes *all* the touch events that occur between the first and last contacts a user makes with the screen.

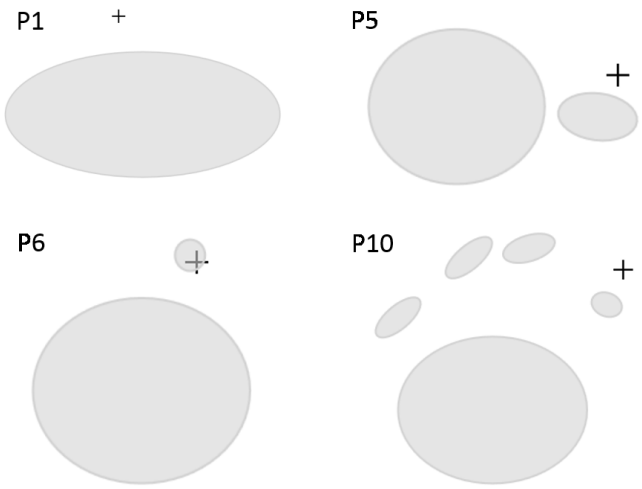

Figure 6. Touch input frames from participants P1, P5, P6, and P10. Ellipses represent contact areas captured by the PixelSense.

Leveraging what we have learned from our exploration of motor-impaired touch, we created *Smart Touch*, a three-step user-specific template-matching algorithm that takes advantage of all the touch information provided in a user's touch process to improve touch accuracy for people with motor impairments.

THE DESIGN OF SMART TOUCH

Smart Touch is a three-step user-specific template-matching algorithm that: (1) analyzes a user's touch process to extract the most relevant touch data, (2) matches that data to previously observed training examples (the templates), and (3) resolves the user's intended (x, y) touch point. Template-matching of touch patterns is employed to match the extracted touch data to previously observed touch instances. The location of the user's intended target is predicted by adding an (x, y) offset to the weighted centroid of the extracted touch data. In this section, we describe the design of our Smart Touch algorithm in detail.

Pose Extraction

The first step in our Smart Touch algorithm is to analyze the user's entire *touch process* to extract the most relevant touch data. The touch process begins when the user first contacts the screen and ends once all registered touches have been removed. The touch process of users with motor impairments is typically longer and includes more touches than the touch process of non-disabled users. Our touch behavior exploration reported above describes why we cannot rely solely on the first and last touch locations as indicators of where the user is intending to touch. Instead, we must rely on all of the information provided throughout the user's touch process. The steps to extract the touch data are explained in more detail below.

Step 1: Deconstruct the Touch Process into Frames

To extract the most relevant touch data, we first deconstruct the touch process into frames (Figure 7). A frame contains all of the captured touch data (*i.e.*, the properties of the ellipses that represent the touches registered by the PixelSense) present at a given time throughout the touch process. A new frame is created every time a new touch is added, lost, or changed. A change event occurs when any of the properties of the ellipse representing the touch is changed, such as the center of the ellipse changing locations.

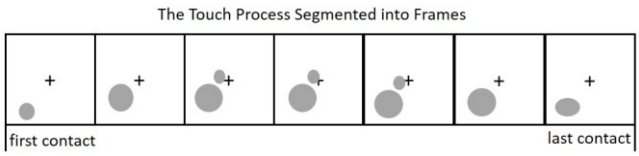

Figure 7. Each frame represents a unique point in time in the user's touch process. Every time a touch is added, changed, or lost, a new frame is created. This figure displays a typical touch process performed by our participants.

Step 2: Evaluate the Stability of Touch Frames

After partitioning the touch process into frames, we must decide on the frame from which to extract the touch data for matching. Based on observations made during our touch behavior exploration, we noticed that many of our participants tended to dwell when they neared their target. We decided that it is at this time that we want to extract the touch data. Dwelling results in neighboring frames having touches with similar properties. We call frames that exhibit small differences from the frames that precedes them *stable*. Conversely, frames that vary greatly from the frames that precede them are considered *unstable*. For a frame to be labeled as "stable," it must be stable in two respects, *movement* and *shape*.

Two stability scores, one each for movement and shape, are assigned to each frame. The movement stability score is assigned based on the difference between the location of the weighted-centroids of the ellipses that represent the touches in the frame being scored and the frame that precedes it. The shape stability score is assigned based on the difference between the sum of the area of the ellipses that represent the touches in the frame being scored and the frame that precedes it. A frame is considered movement- or shape-stable if its movement or shape scores are less than 3% of the sum of all movement and shape scores for all the frames in the touch process, a threshold that we found empirically to work well. A frame that is both movement- and shape-stable is categorized as stable overall (Figure 8).

Step 3: Frame Selection

After all frames have been categorized as stable or unstable, we iterate over all frames to find consecutive stable frames. For each set of consecutive stable frames, we determine their lifespans by subtracting the timestamps of the last and first frames in each set. Touch data is extracted from the frame that occurs halfway (with respect to time) through the life of

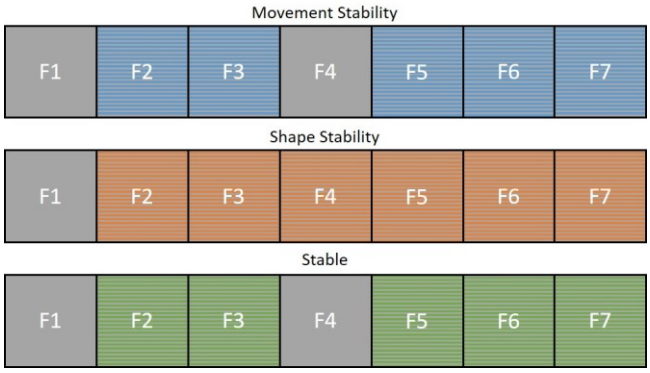

Figure 8. Each of seven frames is labeled stable (highlighted) or unstable (gray) with respect to movement and shape. Frames that are both movement- and shape-stable are categorized as stable overall.

the set of stable frames with the longest lifespan (Figure 9). The extracted touch data is called the *indicative pose*. The touch data in the indicative pose is then sent to a template-matcher to compare the pose to previously observed poses. In the next section, we detail our pose-matching algorithm.

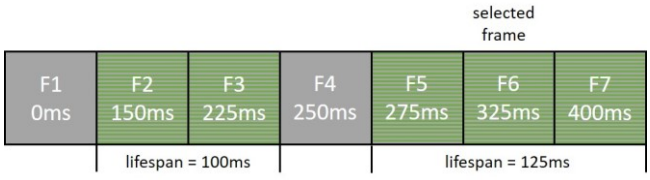

Figure 9. The lifespan of each set of consecutive stable frames is calculated. Touch data is extracted from the frame that is present halfway through the set of stable frames with the longest lifespan.

Pose Matching

Once the touch data has been extracted as the indicative pose for the touch process, we compare it to templates of previously extracted poses (a pose is simply a frame of extracted touch data.) In this section, we detail the steps involved in our template-matching process.

Translating the Pose

All extracted touch data—poses serving as templates and the pose serving as the candidate—are translated to a common reference point. First, a bounding box is fit to the touch data in each pose. The top-left corner of the bounding box is then

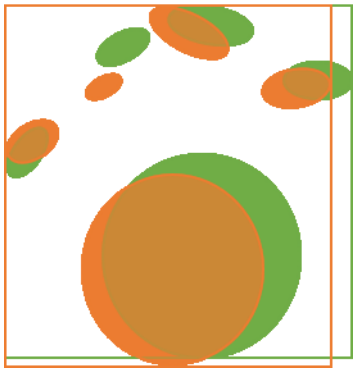

Figure 10. Touch data from two poses translated to the origin, prepared to undergo the template-matching process.

translated to the reference point (0,0). Translation allows poses to be compared regardless of where touches occur on the touch screen (see Figure 10).

Extending $P from Point-Matching to Ellipse-Matching

At this point, the candidate pose *C* and the templates *T* have all been translated. Next, we want to find the template that most closely resembles the candidate. To match a candidate to a template, we extended the $P point-cloud stroke gesture recognizer [36]. The $P recognizer decomposes stroke gestures into point-clouds and uses a nearest-neighbor point-correspondence algorithm to match a candidate gesture to a set of templates. To match the candidate point-cloud *C* to a given template point-cloud *T*, a function *M* matches each point in *C* to exactly one point in *T*. The goodness of *M* is defined as the sum of Euclidean distances for all pairs of points (Eq. 1).

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

Our template-matcher utilizes the underlying behavior of $P to match each of the touch instances (i.e., the ellipses) in *C* to a touch instance in *T*. However, we had to extend the functionality of the $P template-matcher in two ways. First, the template-matcher had to account for the shape properties of the contact areas, which are ellipses. Unlike a point, which only consists of an (x, y) coordinate, an ellipse has a major and minor axis, an orientation, and a center (x, y) location. As a result, our scoring function cannot rely solely on the Euclidean distance between the centers of the ellipses in each pose. To account for the properties of ellipses, we extended $P's Euclidean distance formula: the distance between two ellipses is defined as *the amount of work required to transform one ellipse into another*. Therefore, the distance *D* between 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 [43] 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

$\theta = |(|180 - a_\theta + b_\theta \mid \mathrm{mod}\ 360) - 180|$, where θ is the ellipse orientation in degrees. (See [43] for the origins of this angular distance formula.) Optimizing weights resulted in roughly equal weighting, so for simplicity each w_i was set to 0.25.

The second extension to $P was to allow it to accommodate poses with an unequal number of ellipses. In the original $P recognizer, stroke gestures were resampled so that the candidate stroke and the template strokes always had the same number of points. This resampling procedure allowed each point in the candidate to match exactly one point in the template. Resampling all poses to have the same number of touches would not work in our case, as removing or adding touches would result in a loss of valuable information about a user's touch process. And unlike stroke gestures, our touches do not have a path through space that they necessarily follow. Because we could not guarantee that *C* and *T* would have the same number of ellipses, we devised a recursive variant of $P's point-cloud matching algorithm that allows for an unequal number of ellipses between the candidate and the template. The pseudocode for our recursive $P implementation can be found at the following link: http://bit.ly/10wbHmW. The gist of the algorithm is that ellipses in one pose can serve as the best-matched ellipse for multiple ellipses in the other pose; poses with very different ellipse counts are penalized appropriately; and ellipses that go entirely unmatched also increase distances appropriately.

At the end of the pose-matching process, an *N*-best list is generated and the template with the lowest distance score represents the best match. After the candidate has been matched to a template, the candidate is translated back to its original position on the table. Then Smart Touch predicts the location of the user's intended touch point, described next.

Predicting the User's Intended Touch Point

Embedded in each template is an (x, y) offset representing where the user's intended target was located at the time of the template's creation. (Training examples present a crosshairs so the intended target is known.) The (x, y) offset is encoded as the distance from the weighted centroid of the template's pose to the target's center. This offset is then added to the weighted centroid of the candidate, with the resulting (x, y) coordinate representing Smart Touch's predicted intended touch location.

EVALUATION OF SMART TOUCH

To determine how accurately Smart Touch can predict a user's intended touch point, we performed an evaluation of Smart Touch using the data collected from our 10 participants, described above in our exploration of touch behavior. Smart Touch was used to extract one pose per trial for each trial performed by our 10 participants. As a result, each participant had the same number of poses available for testing as the number of trials he or she completed in our exploration of touch behavior study.

Design and Analysis

We conducted an initial experiment to determine at which number of templates Smart Touch was able to most accurately predict the participants' intended touch location. The initial experiment was a within-subjects design with the following factor and levels:

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

The second experiment compared the best version of Smart Touch from the initial experiment to the land-on and lift-off

locations reported by the PixelSense. The second experiment was a within-subjects design with the following factor and levels:

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

Land-on uses the centroid of the ellipse representing the first touch to contact the screen and Lift-off uses the centroid of the ellipse representing the last touch to leave the screen. Land-on and Lift-off are the selection schemes current touch screens use to determine if a target has been selected, as a user must successfully land *and* lift within a target's bounds. We treated each of these as separate techniques for the sake of measuring distance from an intended touch point (a crosshairs). We did not use area targets (*e.g.*, buttons) as we were interested not in target hit-rates but in distance from a presented (x, y) point.

To conduct our evaluations, we used a testing procedure based on the methods used in machine learning [24] and followed in prior template-matching work [46]. Of the given number of poses *T* performed by each participant (see Table 2), *E* poses were randomly extracted and treated as training examples. *E* was systematically increased from 10 to 40 in steps of 10. Of the remaining $T - E$ poses left, one pose was randomly selected and treated as the candidate. Then it was matched against the library of *E* training examples (of course, the candidate could never serve as a template within the same trial.) For the first experiment, this procedure was performed 100 times for each participant per level of *E*. In total, 10 (participants) × 100 (trials) × 4 (*E* values) = 4000 matching tests were conducted. For the second experiment, this procedure was performed 100 times for each participant, resulting in 10 (participants) × 100 (trials) = 1000 matching tests.

The dependent variable of interest in both experiments was *Error Offset*, measured by the Euclidean distance between the center of the crosshairs in each trial (the actual known target) and the produced (x, y) location reported by Smart Touch in the first experiment, and by each *Selection Technique* in the second experiment. *Error Offset* was analyzed with a mixed-effects model analysis of variance [23]. Our model used fixed-effects for number of training examples *E* in the initial experiment, for *Selection Technique* in the second experiment, and for *Trial* (nested within number of training examples) in both experiments. *Subject* was a random effect to accommodate for repeated measures in both experiments. Although mixed-effects model analyses of variance retain large denominator degrees of freedom, such analyses make detection of significance no easier than traditional fixed-effects ANOVAs.

Results

This section presents the results of our first experiment to see which level of *E* provided the most accurate version of Smart Touch. It also presents results of our second experiment to see how well the best version of Smart Touch could predict the location of our participants' intended touch compared to the native touch sensor's reporting of Land-on and Lift-off.

In the first experiment, mean distances from the intended crosshairs for each level of *E* 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 [18]. 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.

	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 in predicting the location of the participants' intended target.

DISCUSSION

We wanted to discover if our Smart Touch algorithm could improve the touch accuracy of participants with motor impairments using the touch data collected during our touch behavior exploration. Our experiment results show that at 3.01 cm of error offset, Smart Touch is significantly better than both Land-on and Lift-off in predicting the locations of users' intended touch-points. With only 30 templates, which take under 8 minutes to collect, the average Smart Touch error offset across all 10 participants was over three times closer to intended crosshairs than the Land-on and Lift-off techniques. On average, Smart Touch was closer to the crosshairs *for all 10 participants* than either of the two native sensor techniques. Although not all participants were able to complete all 110 trials due to fatigue, we found no indication that fatigue negatively impacted the results of our experiment. The touch behavior of our participants remained quite consistent throughout all of the completed trials.

The average Land-on and Lift-off error offsets of 9.80 cm and 10.65 cm produced by our participants are *far* outside the range of current touch screen widgets. Prior work has found common touch screen widgets to range between 2.6 mm and 4.8 mm [28]. For the majority of our participants, even the recommended widget sizes for users with motor impairments (between 1.2 cm and 1.7 cm [17]) would not be big enough to provide an increase in touch accuracy when using either Land-on or Lift-off. However, with Smart Touch, by increasing accessible widget sizes to 3 cm, 7 of our 10 participants could successfully interact with a tabletop device. And due to the large screen size, tabletop applications could afford to render 3 cm large touch targets.

Our data shows that with 30 templates, we are able to provide significant increases in touch accuracy. Thirty touch examples can be collected from a user in under 8 minutes when he or she first begins to use a touch screen device. As tools and techniques that enable data collection "in the wild" are developed and deployed [11], Smart Touch could unobtrusively collect training examples as users interact with their touch screen devices over time.

Limitations

One limitation of this work is the generalizability of the touch behaviors exhibited by users on the Microsoft PixelSense to other touch screen devices, like phones or tablets, which have smaller form factors than interactive tables. It is possible that the ergonomics of the PixelSense influenced our participants' touch behavior, although we were proactive in preventing any negative effects the PixelSense may have had on our participants' abilities to interact with the screen. Other researchers have noted that users employ similar behaviors across touch screen types, but these researchers did not investigate touch to the extent provided in this paper [2,17,35].

Another limitation is the generalizability of our Smart Touch algorithm to other touch screen devices. The wealth of touch data provided by the PixelSense was the reason it was chosen for the current study. We believe we can achieve similar results on other touch screen devices, such as the Apple iPad, even if the reported touch data is not as complete as what is reported by the PixelSense. Our belief stems from the resilience of our matching algorithm to changes in the weights of Eq. 2. Presumably this means the algorithm can be made effective even if, say, minor axes are not reported for ellipses.

FUTURE WORK

Future studies would include investigating the effectiveness of Smart Touch on touch screen mobile devices, such as smartphones and tablets. It would also be interesting to see how well Smart Touch performs with less accurate touch screen systems. Retrieving accurate touch readings, such as the orientation of a touch or the size of the major and minor axes of the ellipse fitted to the touch contact area, may be more difficult to acquire on certain capacitive-sensing mobile devices. We may, however, be able to utilize other sensor information to improve the Smart Touch algorithm, such as the amount of force applied to the screen.

Another direction of work would test the effectiveness of Smart Touch for non-disabled users experiencing situational impairments [33]. Although Smart Touch was designed for people with motor impairments, the core principle of leveraging users' touch behavior to improve touch accuracy could also be applied to non-disabled users under the effects of a situational impairment, such as while walking.

CONCLUSION

We have explored the touch behavior of 10 people with motor impairments and identified two key behaviors that impact touch screen accessibility. One behavior is that touch screens presume users can interact with the screen using a single fully extended finger. A second behavior is that touch screens presume users possess the fine motor-control skills to both land and lift within the bounds of a target. With Smart Touch, we have made progress on both of these challenges, cutting distances to intended targets by over a third compared to the locations reported by the native touch sensor. In so doing, we have championed ability-based design [42,44], which strives to allow people to use the abilities they have, and make devices and interfaces support those abilities as they are. Smart Touch brings us an important step closer to fully accessible and operable touch screens for all people of all abilities.

ACKNOWLEDGEMENTS

The authors thank Alex I. Jansen, Gabriel Laigo, the staff at Provail, and all of our participants. This work was supported in part by National Science Foundation grants IIS-0952786 and IIS-1217627, and by a Google Research Award. R.-D. Vatavu acknowledges support from the project PN-II-RU-TE-2014-4-1187 financed by UEFISCDI, Romania. Any opinions, findings, conclusions or recommendations expressed in this work are those of the authors and do not necessarily reflect those of any supporter.

REFERENCES

1. Pär-Anders Albinsson and Shumin Zhai. 2003. High precision touch screen interaction. *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '03), ACM Press, 105–112. http://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. *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '13), ACM Press, 1223–1232. http://doi.org/10.1145/2470654.2466158
3. Ravin Balakrishnan. 2004. "Beating" Fitts' law: Virtual enhancements for pointing facilitation. *International Journal of Human-Computer Studies* 61, 6: 857–874. http://doi.org/10.1016/j.ijhcs.2004.09.002
4. Hrvoje Benko, Andrew D. Wilson, and Patrick Baudisch. 2006. Precise selection techniques for multi-touch screens. *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '06), ACM, 1263–1272. http://doi.org/10.1145/1124772.1124963
5. 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. http://doi.org/10.1080/10447318.2011.636294
6. Xiaojun Bi, Yang Li, and Shumin Zhai. 2013. FFitts law: Modeling finger touch with Fitts' law. *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '13), ACM, 1363–1372. http://doi.org/10.1145/2470654.2466180
7. Xiaojun Bi and Shumin Zhai. 2013. Bayesian touch: A statistical criterion of target selection with finger touch. *Proceedings of the ACM Symposium on User Interface Software and Technology* (UIST '13), ACM Press, 51–60. http://doi.org/10.1145/2501988.2502058
8. Karen B. Chen, Anne B. Savage, Amrish O. Chourasia, Douglas A. Wiegmann, and Mary E. Sesto. 2013. Touch screen performance by individuals with and without motor control disabilities. *Applied Ergonomics* 44, 2: 297–302. http://doi.org/10.1016/j.apergo.2012.08.004
9. Morgan Dixon, James Fogarty, and Jacob Wobbrock. 2012. A general-purpose target-aware pointing enhancement using pixel-level analysis of graphical interfaces. *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '12), ACM Press, 3167–3176. http://doi.org/10.1145/2207676.2208734
10. 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. *Proceedings of the Human Factors and Ergonomics Society*, 566–570.
11. Abigail Evans and Jacob Wobbrock. 2012. Taming wild behavior: The input observer for obtaining text entry and mouse pointing measures from everyday computer use. *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '12), ACM, 1947–1956. http://doi.org/10.1145/2207676.2208338
12. 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. *Proceedings of the ACM Symposium on User Interface Software and Technology* (UIST '10), ACM Press, 153–162. http://doi.org/10.1145/1866029.1866055
13. Jon Froehlich, Jacob O. Wobbrock, and Shaun K. Kane. 2007. Barrier pointing: Using physical edges to assist target acquisition on mobile device touch screens. *Proceedings of the ACM SIGACCESS Conference on Computers and Accessibility* (ASSETS '07), ACM Press, 19–26. http://doi.org/10.1145/1296843.1296849
14. Krzysztof Z. Gajos, Jacob O. Wobbrock, and Daniel S. Weld. 2007. Automatically generating user interfaces adapted to users' motor and vision capabilities. *Proceedings of the ACM Symposium on User Interface Software and Technology* (UIST '07), ACM Press, 231–240. http://doi.org/10.1145/1294211.1294253
15. Krzysztof Z. Gajos, Jacob O. Wobbrock, and Daniel S. Weld. 2008. Improving the performance of motor-impaired users with automatically-generated, ability-based interfaces. *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '08), ACM, 1257–1266. http://doi.org/10.1145/1357054.1357250
16. Tiago João Vieira Guerreiro, Hugo Nicolau, Joaquim Jorge, and Daniel Gonçalves. 2010. Assessing mobile touch interfaces for tetraplegics. *Proceedings of the ACM Conference on Human Computer Interaction with Mobile Devices and Services* (MobileHCI '10), ACM Press, 31–34. http://doi.org/10.1145/1851600.1851608
17. Tiago Guerreiro, Hugo Nicolau, Joaquim Jorge, and Daniel Gonçalves. 2010. Towards accessible touch interfaces. *Proceedings of the ACM SIGACCESS Conference on Computers and Accessibility* (ASSETS '10), ACM, 19–26. http://doi.org/10.1145/1878803.1878809
18. Sture Holm. 1979. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics* 6, 2: 65–70.
19. Christian Holz and Patrick Baudisch. 2010. The generalized perceived input point model and how to double touch accuracy by extracting fingerprints. *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '10), ACM, 581–590. http://doi.org/10.1145/1753326.1753413

20. Christian Holz and Patrick Baudisch. 2011. Understanding touch. *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '11), ACM, 2501–2510. http://doi.org/10.1145/1978942.1979308
21. 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. http://doi.org/10.1016/j.apergo.2012.03.003
22. 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. *Proceedings of the ACM SIGACCESS Conference on Computers and Accessibility* (ASSETS '09), ACM, 115–122. http://doi.org/10.1145/1639642.1639663
23. 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.
24. Tom M. Mitchell. 1997. *Machine Learning*. McGraw-Hill Education, New York.
25. Kyle Montague, Vicki L. Hanson, and Andy Cobley. 2012. Designing for individuals: Usable touch-screen interaction through shared user models. *Proceedings of the ACM SIGACCESS Conference on Computers and Accessibility* (ASSETS '12), ACM Press, 151–158. http://doi.org/10.1145/2384916.2384943
26. Kyle Montague, Hugo Nicolau, and Vicki L. Hanson. 2014. Motor-impaired touchscreen interactions in the wild. *Proceedings of the ACM SIGACCESS Conference on Computers and Accessibility* (ASSETS '14), ACM, 123–130. http://doi.org/10.1145/2661334.2661362
27. Maia Naftali and Leah Findlater. 2014. Accessibility in context: Understanding the truly mobile experience of smartphone users with motor impairments. *Proceedings of the ACM SIGACCESS Conference on Computers and Accessibility* (ASSETS '14), ACM Press, 209–216. http://doi.org/10.1145/2661334.2661372
28. Pekka Parhi, Amy K. Karlson, and Benjamin B. Bederson. 2006. Target size study for one-handed thumb use on small touchscreen devices. *Proceedings of the ACM Conference on Human-computer Interaction with Mobile Devices and Services* (MobileHCI '06), ACM Press, 203–210. http://doi.org/10.1145/1152215.1152260
29. R. L. Potter, L. J. Weldon, and B. Shneiderman. 1988. Improving the accuracy of touch screens: An experimental evaluation of three strategies. *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '88), ACM, 27–32. http://doi.org/10.1145/57167.57171
30. Julia Schwarz, Scott Hudson, Jennifer Mankoff, and Andrew D. Wilson. 2010. A framework for robust and flexible handling of inputs with uncertainty. *Proceedings of the ACM Symposium on User Interface Software and Technology* (UIST '10), ACM Press, 47–56. http://doi.org/10.1145/1866029.1866039
31. Julia Schwarz, Jennifer Mankoff, and Scott Hudson. 2011. Monte Carlo methods for managing interactive state, action and feedback under uncertainty. *Proceedings of the ACM Symposium on User Interface Software and Technology* (UIST '11), ACM, 235–244. http://doi.org/10.1145/2047196.2047227
32. Julia Schwarz, Jennifer Mankoff, and Scott E. Hudson. 2015. An architecture for generating interactive feedback in probabilistic user interfaces. *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '15), ACM Press, 2545–2554. http://doi.org/10.1145/2702123.2702228
33. Andrew Sears, Min Lin, Julie Jacko, and Yan Xiao. 2003. When computers fade … pervasive computing and situationally-induced impairments and disabilities. *Proceedings of the 10th International Conference on Human-Computer Interaction* (HCI Int'l' 03), 1298–1302.
34. 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. http://doi.org/10.1016/0020-7373(91)90037-8
35. Shari Trewin, Cal Swart, and Donna Pettick. 2013. Physical accessibility of touchscreen smartphones. *Proceedings of the ACM SIGACCESS Conference on Computers and Accessibility* (ASSETS '13), ACM, 19:1–19:8. http://doi.org/10.1145/2513383.2513446
36. Radu-Daniel Vatavu, Lisa Anthony, and Jacob O. Wobbrock. 2012. Gestures as point clouds: A $P recognizer for user interface prototypes. *Proceedings of the ACM International Conference on Multimodal Interaction* (ICMI '12), ACM, 273–280. http://doi.org/10.1145/2388676.2388732
37. Daniel Vogel and Patrick Baudisch. 2007. Shift: A technique for operating pen-based interfaces using touch. *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '07), ACM, 657–666. http://doi.org/10.1145/1240624.1240727
38. 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. *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '11), ACM Press, 623–626. http://doi.org/10.1145/1978942.1979031
39. Daryl Weir. 2012. Machine learning models for uncertain interaction. *Adjunct Proceedings of the 25th Annual ACM Symposium on User Interface Software*

and Technology (UIST '12), ACM, 31–34. http://doi.org/10.1145/2380296.2380313

40. Daryl Weir, Simon Rogers, Roderick Murray-Smith, and Markus Löchtefeld. 2012. A user-specific machine learning approach for improving touch accuracy on mobile devices. *Proceedings of the ACM Symposium on User Interface Software and Technology* (UIST '12), ACM Press, 465–476. http://doi.org/10.1145/2380116.2380175
41. Daniel Wigdor, Clifton Forlines, Patrick Baudisch, John Barnwell, and Chia Shen. 2007. Lucid Touch: A see-through mobile device. *Proceedings of the ACM Symposium on User Interface Software and Technology* (UIST '07), ACM Press, 269–278. http://doi.org/10.1145/1294211.1294259
42. 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.
43. 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. *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '09), ACM, 1401–1410. http://doi.org/10.1145/1518701.1518912
44. 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. http://doi.org/10.1145/1952383.1952384
45. 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. *Proceedings of the ACM Symposium on User Interface Software and Technology* (UIST '03), ACM Press, 61–70. http://doi.org/10.1145/964696.964703
46. Jacob O. Wobbrock, Andrew D. Wilson, and Yang Li. 2007. Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes. *Proceedings of the ACM Symposium on User Interface Software and Technology* (UIST '07), ACM, 159–168. http://doi.org/10.1145/1294211.1294238