PersonalTouch: Improving Touchscreen Usability by Personalizing Accessibility Settings based on Individual User's Touchscreen Interaction

Yi-Hao Peng
National Taiwan University
b03902097@ntu.edu.tw

Muh-Tarng Lin
National Taiwan University
tommy60703@gmail.com

Yi Chen
National Taiwan University
r07922059@ntu.edu.tw

TzuChuan Chen
National Taiwan University
aldrich1221@gmail.com

Pin Sung Ku
National Taiwan University
scott201222@gmail.com

Paul Taele
Texas A&M University
ptaele@cse.tamu.edu

Chin Guan Lim
National Taiwan University
sky010flyer@gmail.com

Mike Y. Chen
National Taiwan University
mikechen@csie.ntu.edu.tw

ABSTRACT

Modern touchscreen devices have recently introduced customizable touchscreen settings to improve accessibility for users with motor impairments. For example, iOS 10 introduced the following four Touch Accommodation settings [10]: 1) Hold Duration, 2) Ignore Repeat, 3) Tap Assistance, and 4) Tap Assistance Gesture Delay. These four independent settings lead to a total of more than 1 million possible configurations, making it impractical to manually determine the optimal settings. We present PersonalTouch, which collects and analyzes touchscreen gestures performed by individual users, and recommends personalized, optimal touchscreen accessibility settings. Results from our user study show that PersonalTouch significantly improves touch input success rate for users with motor impairments (20.2%, N=12, p=.00054) and for users without motor impairments (1.28%, N=12, p=.032).

CCS CONCEPTS

• **Human-centered computing** → *Accessibility design and evaluation methods*; *Human computer interaction (HCI).*

KEYWORDS

Accessibility; Motor Impairment; Personalization; Touchscreen Interaction; Gesture Recognizer

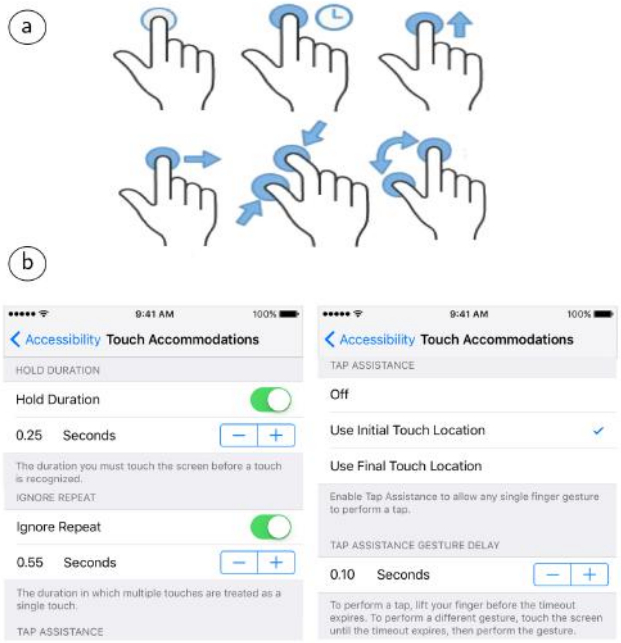

Figure 1: (a) PersonalTouch first collects touchscreen gestures, and then recommends personalized, optimal accessibility settings. (b) PersonalTouch displays the iOS accessibility settings for a 72-year old user with mild tremors, which improved the user's touchscreen input success rate from 48.4% to 66.3%.

CHI 2019, May 4–9, 2019, Glasgow, Scotland UK

ACM ISBN 978-1-4503-5970-2/19/05…$15.00
https://doi.org/10.1145/3290605.3300913

ACM Reference Format:
Yi-Hao Peng, Muh-Tarng Lin, Yi Chen, TzuChuan Chen, Pin Sung Ku, Paul Taele, Chin Guan Lim, and Mike Y. Chen. 2019. PersonalTouch: Improving Touchscreen Usability by Personalizing Accessibility Settings based on Individual User's Touchscreen Interaction. In *CHI Conference on Human Factors in Computing Systems Proceedings (CHI 2019), May 4–9, 2019, Glasgow, Scotland UK*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3290605.3300913

1 INTRODUCTION

Touchscreen devices have gesture recognizers that are constantly interpreting users' touches as one of the supported gestures, such as *tap*, *scroll*, *swipe*, *long press*, *rotate*, and *pinch*. These gesture recognizers have default settings that are optimized for users with good dexterity; however, they perform poorly for users with motor impairment in controlled settings [14, 17, 22, 24, 31, 39, 41, 44] and in the wild [2, 32, 34]. To improve touchscreen accessibility, iOS and Android have recently introduced accessibility settings to allow the customization of these gesture recognizers. For example, Apple introduced Touch Accommodation settings in iOS 10 [10]: 1) *Hold Duration*, 2) *Ignore Repeat*, 3) *Tap Assistance*, and 4) *Tap Assistance Gesture Delay*.

Past research has shown that accessibility settings are difficult to discover, access, and understand [2, 41]. Touchscreen settings are especially difficult for users to try and evaluate, because users must test each setting across all types of gestures and each type of gesture is affected differently. Furthermore, gesture recognizer settings require one or more timing thresholds to be configured correctly. Using iOS Touch Accommodations as an example, these four independent touchscreen settings have a total of 1.15 million possible configurations, which makes it impossible for users to configure optimally.

This paper presents PersonalTouch, which improves touchscreen accessibility by first collecting and analyzing individual users' touchscreen input, and then recommending personalized, optimal accessibility settings. Our current system first utilizes an iOS app to collect touch input, then analyzes how well various types of the user's gestures are recognized using all 1.15 million configurations, and finally recommends the optimal settings for iOS Touch Accommodations.

To evaluate our approach, we collected data from 12 participants with motor impairments, including spinal cord injury, cerebral palsy, Parkinson's disease, and mild tremors. We also collected data from 12 participants without motor impairments. Study results show that PersonalTouch significantly improved touch input success rate for all users with motor impairments, with an average improvement of 20.2% (N=12, $p=.00054 < .05$). PersonalTouch also improved the input success rate for 50% of the users without motor impairments, with an average improvement of 1.28% (N=12, $p=.032 < .05$).

2 RELATED WORK

The most relevant prior work have: 1) improved keyboard and mouse accuracy by optimizing accessibility settings, and 2) improved touchscreen accessibility for people with motor impairments through novel gestures, interfaces, and recognizer algorithms.

Keyboard and Mouse Accessibility Settings

Prior work on optimizing accessibility settings provided by operating systems has focused on improving the accessibility of keyboard and mouse. Dynamic Keyboard [40] proposed an approach to analyze a user's keyboard use and optimize keyboard accessibility settings, including Key Repeat Delay, Key Repeat Rate, and Bounce Keys. Koester et al. [26–28] developed software agents that processed a user's keyboard and mouse input, and then optimized the corresponding accessibility settings including Double-click Time, Double-click Distance and Pointer Speed for mouse; and Key Repeat Delay, Key Repeat Rate, and Sticky Keys for keyboard. Our work is the first investigation into optimizing touchscreen accessibility settings.

Touchscreen Accessibility

Extensive research has been conducted to understand the difficulties users with motor impairments encounter when using touchscreens [2, 14, 17, 22, 24, 31, 32, 34, 39, 41, 44]. Various approaches have been proposed to improve touchscreen accessibility, including novel gestures and novel user interfaces. For novel gestures, swabbing has been shown to improve target selection for users with hand tremors [30, 43]. For novel user interfaces, a two-stage selection interface [46] and a customized adaptive keyboard layout [38] have been proposed. These approaches require users to learn new gestures and new interfaces, whereas our approach does not require users to change their behavior.

Algorithms have also been developed for users with motor impairments to improve tapping recognition and to also improve the prediction of the (x,y) coordinates of a tap, without requiring users to modify their existing behavior. Montague et al. [32] trained user- and session-specific tapping and swiping gesture models. Mott et al. [33] developed a template matching-based algorithm that is robust to multiple concurrent touches. Our work is complementary to advancements in gesture recognizer algorithms in that PersonalTouch evaluates all possible accessibility configurations and recommends the optimal settings. Furthermore, in addition to tapping performance, we evaluate how these configurations perform for all types of touch gestures.

3 SYSTEM DESIGN AND IMPLEMENTATION

In order to recommend personalized, optimal settings, our system needs to: 1) support the collection of touch data for all types of touchscreen gestures, and 2) evaluate all possible touchscreen accessibility configurations.

Touch Input Tasks

Our tasks covered all six types of the standard touchscreen gestures supported by Android and iOS [4, 18]: tap, long

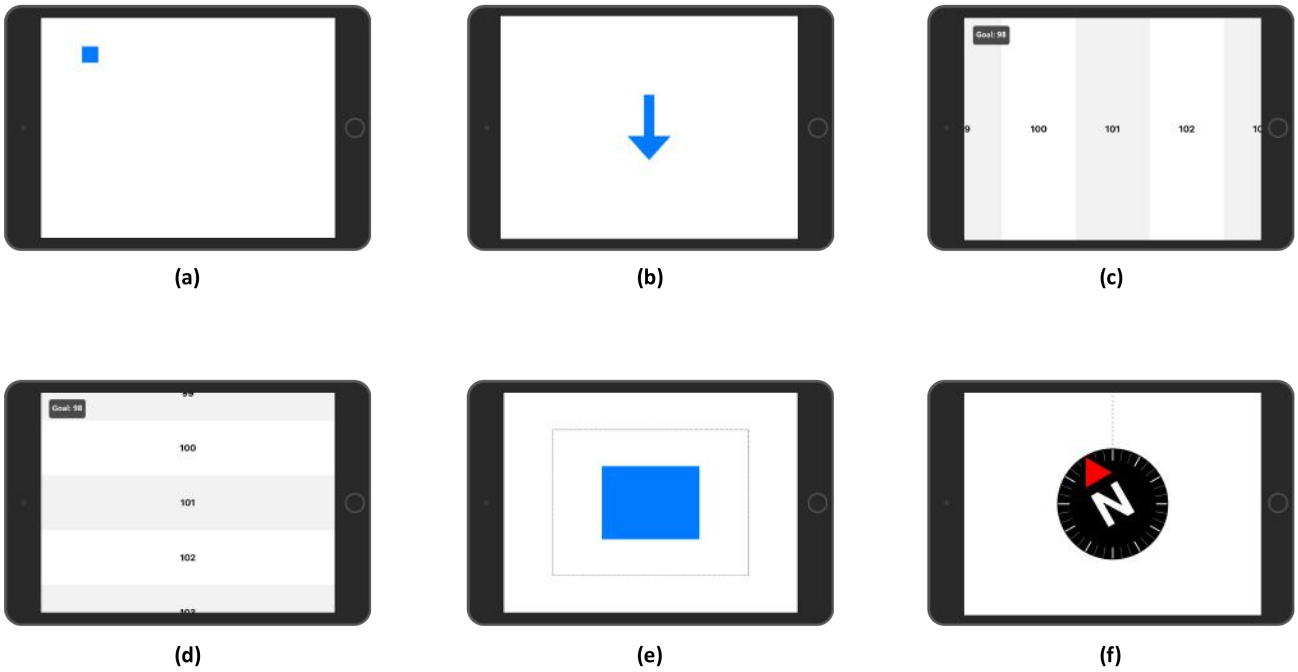

Figure 2: Touch input tasks for collecting individual touch data. The tasks are designed to collect each of the 6 standard gestures supported by Android and iOS: (a) *tap* and *long press*, (b) *swipe*, (c) *horizontal scroll*, (d) *vertical scroll*, (e) *pinch*, and (f) *rotate*.

press, swipe, scroll, pinch, and rotate. These gestures can be categorized as: 1) *discrete* gestures: tap, long press, and swipe, which are recognized after users complete the entire gesture; and 2) *continuous* gestures: scroll, pinch, and rotate, which are continuously tracked by the system with the user interface continuously updated throughout the gesture. We designed the gesture task user interfaces based on designs from [17, 22, 36], and the application user interfaces based on designs from the stock iOS and Android apps [3, 5, 19, 20]. Our app is developed using Swift 4.1 and Xcode 9.3, using iOS UIkit [9] and ResearchKit [6].

Tap. This gesture is the most frequently used touch gesture. For this input interface design, we adapted the tap task from [22] by dividing the screen into a 5×5 grid, where the center of each grid contained a blue square button, and showing one target per trial (Figure 2.a). The target sizes are 44pt and 80pt for the minimum sizes of buttons and icons, respectively, as specified in Apple's iOS Human Interface Guidelines [8].

Long Press. This gesture is used by applications for activating common various tasks, such as copy-pasting text and repositioning mobile app icons. For this input task design, we chose the same target size as the tap task (shown in Figure 2.a), but instead divided the screen into nine regions to reduce the number of trials.

Swipe. This gesture involves performing directional gestures that satisfies specified thresholds [7]. The requirements and discreteness for executing the gesture makes it infrequently used in a non-gaming app's functionality, due to its discontinuous feedback. On the other hand, this touch gesture is widely adopted in popular gaming app such as *Temple Run* and *Minion Rush*, where users can control their in-game character's direction through swiping actions. For this input task design, we adapt the swipe tasks conducted in [22] and simplified it into a whole-screen swipe task in four directions, matching the directions supported by iOS and Android's built-in gesture recognizers (shown in Figure 2.b).

Scroll. This gesture is also known as *pan*. Similar to swipe, this gesture is a directional gesture, but is continuous rather than discrete. This gesture is widely used in any scrollable views, and in place of swipe for most non-gaming apps. Unlike swipe, scroll does not prompt users to perform accurate direction gesturing at a minimum speed, but instead moves the target interface at the speed of the user's corresponding touch movements. For this input task design, we adapted the scroll tasks from prior works [12, 35, 42] and from the

scrolling interaction for navigating the iPad tablet's Apple App store layout [3]. We display five numbered blocks on the screen at a time, with three blocks displayed in the middle and the remaining two at the borders (shown in Figure 2.c and 2.d). The user's goal is to scroll through the target numbered blocks that are prompted at the top-left portion of the screen into the screen view. We focus on two scrolling directions for the tasks: vertical and horizontal. We also focus on two distinct target distances from the screen's border that could be easily scrolled into the screen view from our original design: near-distance and far-distance. Users are prompted in the task to make a best single attempt at scrolling through the target block into the screen view.

Pinch. This gesture is a multi-touch gesture for zooming, commonly used in apps such as Google Maps [19] and Apple Photos [5]. For this input task design, we adapted the pinch tasks from prior works [13, 15, 25]. Our pinch task consists of display a rectangle of size proportional to one-ninth of the screen's dimensions onto the user's view. Users are prompted to pinch-to-resize the displayed rectangle to a dotted-line target in a single attempt (shown in Figure 2.e). We selected four different sizes for the users to pinch-to-resize relative to the original target's size: double, quadruple, one-half, and one-quarter.

Rotate. This gesture is a multi-touch gesture that is used in apps such as Apple Photos [5] and Google Maps [19]. For this input task design, we modified the rotation tasks from [19, 23, 45]. Users are shown an on-screen compass with a default turn direction during the rotation task, and can manipulate the compass by rotating the whole view. The goal of this task is to rotate the compass to the top-right direction, which is marked with a dotted line (shown in Figure 2.f). Possible angles between the default and top-right direction are 30 degrees and 60 degrees, along with two possible rotation directions of clockwise and counterclockwise.

Recommending Personalized Settings

As users perform the touch tasks, our app synchronizes the touch input data to our cloud-based backend for processing. Our recommendation engine performs the following three phases of computation:

(1) **Settings Simulation:** for all possible combinations of accessibility settings, compute how each combination of settings modify the raw touch input data.
(2) **Input Success Rate Calculation:** for each modified touch input data from 1), compute the gesture recognition success rate, so that we have the success rate for each of the 6 types of gestures.
(3) **Settings Recommendation:** compute the most optimal combination of settings by weighting the gesture recogniztion success rates with the expected gesture ratios.

Settings Simulation. Our current system focuses on iOS' Touch Accommodations accessibility settings introduced in iOS 10 [10]. There are four such configurable settings, described below, with all time-related settings being configurable from 0.10s to 4.00s in increments of 0.05s.

(1) **Hold duration:** sets the device to respond to touches only after the user holds their finger on the screen for the specified period of time.
(2) **Ignore repeat:** sets the period of time that the device will treat several touches as one. This should be turned on when users have trouble touching the screen just once.
(3) **Tap assistance:** sets the device to respond to the first or the last place a user touches. If users touch the screen at the place they want, but their fingers drag to a different place before they can make a selection, then *Tap Assistance* with *Use Initial Touch Location* should be turned on. If users have trouble touching the screen at the place they want, but they can drag their fingers to the intended location, then *Tap Assistance* with *Use Final Touch Location* should be turned on.
(4) **Tap assistance gesture delay:** With *Tap Assistance*, the device responds to a tap when users lift their fingers within a certain period of time, called the *gesture delay*. The device can respond to other gestures, like scrolls, if the users wait longer than the gesture delay.

One consideration worth mentioning is that these settings affect the behavior of each other. For example, if *Tap Assistance* is set to the initial location and *Hold Duration* is activated, there will be two timers set on the first point the user touches: *Hold Duration* timer followed by the *Tap Assistance Gesture Delay* timer. At this stage, we enumerate all possible combinations of settings and compute how a user's raw touch data is modified by each combination of settings. We then compute the input success rate using these modified touch data.

Input Success Rate Calculation. To evaluate how an accessibility configuration affects input success rate, we need to be able to run the touch data through the device's built-in recognizers. However, because iOS does not allow the thresholds of the built-in recognizers to be changed via any API, we needed to simulate the exact behavior of the built-in recognizers to compute how the modified touch data will be interpreted by the recognizers.

To better understand how the system recognizers interpret touch events, we investigated public and private iOS APIs to determine the essential thresholds used by the different iOS gesture recognizers [4]. Specifically, we reverse

engineered iOS gesture recognizers using the following two steps: 1) Apple's developer docs describing the factors each recognizer uses for interpreting gestures, and 2) run-time introspection of the names/values of parameters and constants using Xcode's debugger via key-value coding. These two steps provided sufficient information to implement *tap*, *long press*, and *pan* recognizers exactly as iOS. For *swipe*, *pinch*, and *rotate*, we had to disassemble and debug using Hopper [11] in order to trace the assembly code to understand the algorithms, variables, and constants.

To validate our recognizers, we used the data set from our 24-person study mentioned in the next section of the paper, which had a total of 108×24=2,592 gesture trials. In addition, we developed a second data collection app to collect additional gesture trials. 10 participants (mean age=25.6, 4 female) were asked to directly perform each of the six types of gestures at 100 times each, for a total of 6,000 gesture trials. Validating these 8,592 gestures against iOS showed that we were able to recognize the gesture events correctly for all gesture types. For continuous gestures that also reported numerical values (e.g., rotation angle and scale ratio), we had <1% error between the reported numerical values.

After calculating the success rate for each given gesture using all possible settings, we then conduct several comparisons to identify the most optimal setting.

Settings Recommendation. By changing the parameters of our gesture recognizers, we can calculate the success rate for each type of gestures for all possible setting configurations. To calculate the overall success rate for a configuration, we multiply the gesture success rates by the ratio of the gestures. This ratio of gestures varies depending what apps and tasks that users are performing. Unfortunately, we have not been able to find a references on a real-world ratio of gestures. As a result, we propose and consider two different ratios of gestures in the calculations of a configuration's overall success rate. The first set, *Study*, sets the ratio of each gesture task to their corresponding number of trials performed for that particular gesture task, as detailed in Section 4: Procedure. The second set, *Uniform*, equally distributes the ratios of gestures regardless of their usage frequency, such that more commonly-performed gestures are set with the same ratio as less commonly-performed gestures.

For the rest of the paper, we will investigate these ratios of gestures for evaluating input success rate. The exact ratios for each set correspond to the gesture tasks of: a) *tap*, b) *long press*, c) *swipe*, d) *horizontal scroll*, e) *vertical scroll*, f) *pinch*, and g) *rotate*.

- ***Study* Gesture Ratio:** {50:18:8:8:8:8:8}
- ***Uniform* Gesture Ratio:** {1:1:1:1:1:1:1}

To choose the optimal configuration, we used the same cross-validation methodology used by prior works [29, 33] in gesture recognition to train and test on separate gesture trials. Specifically, we conducted 5-fold cross-validation at 10 times each for each participant. We first randomly partitioned the 108 gesture trials into a collection of five sets that are nearly equal in count, {22, 22, 22, 21, 21}, with a balanced sampling for each gesture. We then used four sets to train the optimal settings and tested on the remaining one set. The 5-fold validation was repeated 10 times, for a total of 50 runs, where the success rate was then averaged. During the process, we first sort all configurations by overall success rate after applying the specific gesture ratio, and then by the total time duration. If multiple configurations have the same success rate, then the configuration that is the most responsive (i.e., with the least wait time) is preferred as the optimal one. After ten runs, the most selected setting will be the suggested optimal setting.

System Performance. Our current recommendation engine is written using the CUDA toolkit [37] to utilize GPU acceleration, and is hosted on the GPU Accelerated Computing instances on Amazon's Elastic Computing Cloud (EC2) [1] It

Currently, calculating the success rates for a given gesture using all 1.15 million possible settings takes 2.78s per gesture. This only has to computed once for each gesture used in the training set, and could be optimized using pruning strategies. For example, stopping evaluating longer duration settings once it finds a duration setting that results in no touch events. Analysis shows that this pruning strategy would eliminate an average of 40% of the possible settings.

Identifying the most optimal settings given a gesture training set and a target gesture ratio currently takes 0.01s. It is fast enough for real-time adaptation of changing the target gesture ratio (e.g. switching apps or having a sliding window of recently used gestures).

4 USER STUDY

The user study aims to quantify how PersonalTouch affects input performance compared to the default touchscreen configuration, which has all accessibility settings turned off. Moreover, the study explores how participants with motor impairments currently use touchscreen accessibility features.

Participants

We recruited 12 participants (age 22 to 80, mean=59.2, SD=23.0, 8 females) with diagnosed and self-reported motor impairments as described in Table 1. We also recruited 12 participants without motor impairments (age 22 to 70, mean=32.4, SD=17.3, 6 females). All participants used touchscreen devices regularly, and also used their fingers for touchscreen input except for P1, whom regularly used a stylus with her mouth.

					Self-Reported Impairments										
ID	Age	Gender	Device	Health Condition	Fa	Co	St	Mo	Gr	Ho	Tr	Sp	Se	Dir	Dist
P1	41	F	Smart phone	Spinal cord injury	■			■	■	■				■	■
P2	27	F	Smart phone	Spinal cord injury	■		■	■	■	■			■		
P3	22	F	Smart phone/Tablet	Cerebral palsy	■	■		■	■	■					
P4	25	F	Smart phone	Cerebral palsy	■		■		■				■	■	■
P5	74	M	Tablet	Parkinson's	■	■	■	■	■	■	■				
P6	76	M	Smart phone/Tablet	Parkinson's	■	■		■	■	■	■				
P7	77	F	Tablet	Parkinson's	■	■		■			■				
P8	80	M	Tablet	Parkinson's	■	■	■				■			■	■
P9	73	F	Tablet	–							■				
P10	72	F	Smart phone/Tablet	–		■					■				
P11	73	M	Tablet	–							■				
P12	70	F	Smart Phone/Tablet	–							■				

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

Table 1: Details for the motor impaired group, including gender, age, regularly-used device, and health condition. Categories of self-reported impairments are from Findlater et al. [16].

Procedure

We first interviewed participants about their touchscreen experience and how they currently use accessibility features. Participants were seated comfortably at a desk in a quiet office setting, with a 12.9 inch iPad Pro running iOS 11.2 placed on the desk. We then explained the iOS Touch Accommodations settings [10] and observed how they would configure them.

Participants then practiced and performed the gesture tasks in a fixed sequence of: *tap*, *long press*, *swipe*, *horizontal scroll*, *vertical scroll*, *pinch*, and *rotate* (shown in Figure 2). Each trial began when the first touch event was registered, and ended after no touch event occurred for one second. This one-second sliding window is necessary to record unintended touches such as rapid touches, due to Parkinson's disease or additional fingers; and palm resting, due to users' natural inclination of resting their palms on the touchscreen.

Each participant performed 108 gesture trials: 50 (tap) + 18 (long press) + 8 (swipe) + 8 (horizontal scroll) + 8 (vertical scroll) + 8 (pinch) + 8 (rotate). The total number of trials is

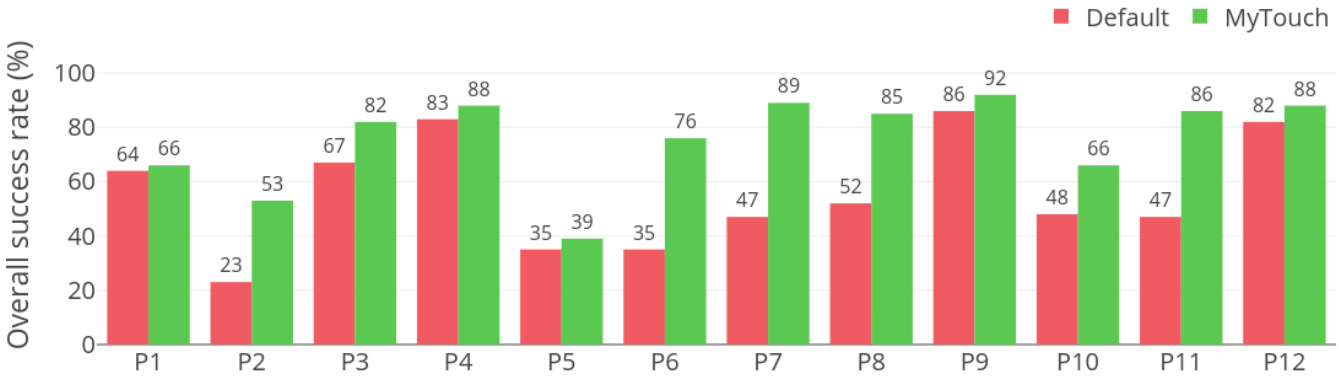

Figure 3: Touchscreen input success rate (%) for the 12 participants with motor impairments when using the default touchscreen settings versus PersonalTouch.

capped to about 100 to ensure that participants would not be exhausted, and the number of trials per type of gesture is chosen to reflect their relative usage. The study session took about 1 hour for users with motor impairments and about 30 minutes for users without motor impairments.

5 RESULTS

Our user study includes qualitative feedback on accessibility features and quantitative results on improved touchscreen input performance.

Awareness and Usage of Accessibility Features

3 of the 12 participants with motor impairments have used at least one iOS accessibility feature. All 3 have used *Assistive Touch* as a shortcut to switch between applications and to control the volume. 6 participants were aware of accessibility features but have not used any, while the remaining 3 were not aware of them. Only 1 participant, P4, knew about Touch Accommodations before our study because her doctor introduced her to these functions. However, P4 did not use them because she did not know how to adjust these settings by herself. After explaining the customizable Touch Accommodations settings to the participants, all expressed that they were uncertain how to choose the appropriate settings.

Participants with Motor Impairments

Figure 3 shows the input success rate for the 12 participants with motor impairments for tasks with our study-specific gesture ratio (see Section 3). The average success rate significantly improved by 20.2% (pairwise t-test: N=12, p=.00054 < .05), from 55.7% using the default settings to 75.9% using PersonalTouch.

Participants who currently have more difficulty using touchscreens had larger improvement with PersonalTouch. The average absolute improvement was 29.1% versus 11.2% for participants with ≤ 50% initial success rate versus participants with > 50% initial success rate, respectively.

Our suggestion also show improvements on tasks with other gesture ratio. For the tasks with equal ratio of all types of gestures, the success rate significantly improved by 8.5% (pairwise t-test: N=12, p=.00028 < .05), from 64.5% using the default settings to 73.0% using PersonalTouch.

Table 2 shows the optimal, personalized configurations for the two different input scenarios that that we earlier proposed for our work (see Section 3): tasks with equal ratio values for all the gesture types (i.e., *Uniform* gesture ratio), and tasks with our study-specific gesture ratio (i.e., *Study* gesture ratio). PersonalTouch recommends different settings for different participants because it is based on user-specific models. Table 2 also shows that participants with severe motor impairments tend to need longer time thresholds compared with people with minor impairments. In addition, depending on the ratio of the different types of gestures, PersonalTouch may recommend different accessibility settings for the same user in order to optimize for the overall success rate.

Participants without Motor Impairments

PersonalTouch recommended the default settings as the optimal settings for 6 of the 12 participants without motor impairments. For the other 6 participants, there is a slight—but statistically significant—improvement using PersonalTouch. Overall, the success rate improved by 1.28% (pairwise t-test: N=12, p=.032 < .05), from 95.40% to 96.68% using the study-specific gesture ratio.

Our analysis found that the errors are primarily due to tapping being interpreted as scrolling due to unintentional lateral movement before lifting the fingers, where PersonalTouch recommends turning on Tap Assistance with a short 0.10s duration to mitigate this. Based on our findings, PersonalTouch improves touchscreen usability for users with a wide range of motor abilities.

Breakdown Analysis for Different Gestures

We also found that when optimizing for the overall recognition rate given a gesture ratio, the most frequently-used gesture type would have the largest improvement, while less frequently-used gesture types would either have smaller improvements or even reduced recognition rates. The following list shows all the gesture recognition rates using PersonalTouch with the study-specific gesture ratio, where *tap* showed the largest improvement while the least frequently-used gestures—*pinch* and *rotate*—actually decreased in recognition rate.

- **Tap:** 46.3% → 85.5% (+39.2%)
- **Long Press:** 45.8% → 55.1% (+9.3%)
- **Swipe:** 82.3% → 90.1% (+7.8%)
- **Scroll:** 86.1% → 92.2% (+6.1%)
- **Pinch:** 47.2% → 38.1% (-9.1%)
- **Rotate:** 56.6% → 46.8% (-9.8%)

6 DISCUSSION

App-specific Optimization

Different apps have a different mix of gesture ratios. For example, messaging apps use more tapping and vertical scrolling, compared to photo album apps that use more horizontal scrolling. In addition to our current system-wide optimization, we can further improve accessibility by analyzing app-specific gesture ratios and dynamically adjust accessibility settings based on the current app.

In general, the task weights could be modified through micro factors: the application that people used and the condition (e.g., sit, walk) in which they used the app; or by macro factors: the most-used input method for each individual user

	Tasks w/ Equal Ratio of All Types of Gestures			Tasks w/ Study-Specific Ratio		
ID	Hold Duration	Ignore Repeat	Tap Assistance	Hold Duration	Ignore Repeat	Tap Assistance
P1	-	-	INIT(0.10)	-	-	INIT(0.10)
P2	-	0.10	INIT(0.25)	-	0.10	INIT(0.25)
P3	-	2.30	INIT(0.25)	-	2.15	INIT(0.35)
P4	-	1.20	-	-	0.80	-
P5	-	0.70	-	0.30	0.45	INIT(0.20)
P6	-	-	INIT(0.25)	-	-	INIT(0.25)
P7	-	0.15	INIT(0.50)	-	0.10	INIT(0.55)
P8	0.15	-	INIT(0.25)	0.20	-	INIT(0.25)
P9	-	-	INIT(0.15)	-	-	INIT(0.15)
P10	0.20	0.75	-	0.25	0.55	INIT(0.10)
P11	0.10	-	INIT(0.15)	0.10	-	INIT(0.15)
P12	-	-	INIT(0.15)	-	-	INIT(0.20)

Legend: INIT/FINAL= Tap Assistance using initial/final touch location. (UNIT: seconds)

Table 2: Optimal iOS Touch Accommodations settings for the 12 participants with motor impairments, showing that the optimal settings differ for: (a) tasks with uniform distribution of all types of gestures versus (b) tasks with our study-specific gesture ratio.

and different personal touchscreen behavior along with the growth of age. With dynamic task weights, our suggested settings could better personalize and accommodate to the user's individualized needs.

Suggested Configuration on Settings beyond Touch Accommodations

The suggested configurations that our method provided are not limited to Touch Accommodations. The input gestures mapping is one common method that is applied in several accessibility settings including *Switch control*, which provides an alternative way to interact with the application by using the universal switch method that people are most comfortable with; and *Assistive touch*, with customized gestures that map the single touch gestures to multi-touch gestures such as *pinch* and *rotate*. For example, P1 was not sure as to whether she needed to turn on the settings that could use single finger to pinch and to rotate. After running the analysis on her behavior data, we discovered that she performed well on the tap task and scroll task, while had difficulties performing multi-touch gesture. Since P1 used a stylus as her chosen input medium, it was not easy for her to complete those challenging tasks. We provided her with suggestions on using this specific accessibility feature when she need to pinch or to rotate on the screen, and received positive feedback from her such as, "Now I can use the map more easily."

Contribution to People with Different Impairments and Who Use Other Devices

Our method could also be applied to people with different difficulties for touchscreen input. For users with visual impairments as an example, such users often used an accessibility feature called *VoiceOver*, which can be combined with hand gestures to access information on touchscreen devices. According to other previous studies [41] though, some people with both visual and physical impairments had difficulty performing some of the default controlled gestures—such as *three-finger swipe*—for other accessibility features. These particular users can benefit from our evaluation process for touchscreen input, providing them with the most suitable input method through better combinations with other accessibility features.

Furthermore, our method is not only limited to iOS devices, and can potentially be adapted to Android devices that incorporate more advanced customizable settings for accessibility. One example is an accessibility feature called *Touch and hold delay* [21], which sets the time threshold for distinguishing *tap* and *long press*. Our methods could not only be customized on some Android devices, but could also

provide the suggested configuration on such a feature by analyzing the user's touch gesture data.

As for the applicability on devices with smaller screens (e.g., smartphones), we could generalize the training data collected from tablets onto smartphones by sampling only tasks that also fit on a smartphone screen for training purposes. For example, the center 3×3 grid of our 5×5 grid of *tap* targets on an iPad actually falls within the screen size of a large iPhone, and the training data can be generalized as such. Actual data collection on smartphones would be needed to validate how well this approach works, and also how well the optimal settings for a tablet generalizes to a smartphone—both of which we would like to explore in a future study.

7 LIMITATIONS AND FUTURE WORK

Our vision of PersonalTouch is to dynamically adapt to users by monitoring their individual touchscreen interactions. The continuous observations and data collections on the user's touch input is also crucial for the dynamic weight. However, there were limitations of our approach that we discovered, and are viable areas for investigating as ideal next steps.

Initial Hand Positions for Gesturing

Our gesture tasks were designed to have a short rest between each trial for the participants to reset their hand position, in order to control for the effect of the proximity of the previous target. However, users perform real-world gesturing tasks on their touchscreen devices that do not assume that they always reset the position of their hands each time they perform a different gesture. This discrepancy between our study requiring that participants reset their hand positions and real-world tasks that do not require such an assumption may affect the calculation of our recommendations. As such, we wish to pursue further study that better understands these differences and what impacts may occur from them.

Broader Demographics of Study Participants

Another future work area involves further expanding our current size of participants in our study to include more diverse demographics. That is, we are interested in conducting user studies that evaluate our method more deeply towards people with difficulties in touchscreen interactions. Furthermore, we are also interested in broadening the conditions of our user study, which also emphasizes that participants interact on a wider range of device system settings and sizes. By investigating our approach on these additional populations and conditions, we expect to see general improvements on touchscreen usability and accessibility for a wider range of interaction behaviors.

Real-world Frequency of Gesture Tasks

One other limitation of our work is in the artificial selection of our ratio values for our gesture tasks. In particular, we explored gesture ratios that focused on two different sets: one where each gesture ratio corresponded to the number of trials of their gesture task in our user study, and one where all the gesture ratios were set uniformly equivalent to each other. Since the ratio values of the gestures not only varies depending what apps and tasks users are performing, but also affect the overall success rate of a configuration, we would like to select gesture ratio values that are more closely aligned with gesture task usage in real-world touchscreen interactions. To do so, we would like to conduct user studies that also investigate the frequency of occurrence for each gesture task on different popular mobile apps and common activities associated with those apps.

8 CONCLUSION

We present PersonalTouch, our approach to improve touchscreen usability by suggesting personalized, optimal accessibility settings. Our study results demonstrated that PersonalTouch significantly improved users' touch input success rate on our gesture tasks, with 20.2% improvement for all our users with motor impairments and 1.28% improvement for half our users without motor impairments. We believe that PersonalTouch can better accommodate users in more optimally setting their touch input configurations, and therefore better enhancing their touchscreen gesture interaction experiences.

REFERENCES

[1] Amazon. 2018. Amazon EC2 service instance. https://aws.amazon.com/ec2/instance-types/p2

[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 SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 1223–1232.

[3] Apple. 2018. Apple App Store. https://www.apple.com/ios/app-store/

[4] Apple. 2018. Apple Gesture Recognizer. https://developer.apple.com/documentation/uikit/uigesturerecognizer

[5] Apple. 2018. Apple Photo. https://www.apple.com/ios/photos/

[6] Apple. 2018. Apple Researchkit. https://www.apple.com/researchkit/

[7] Apple. 2018. Apple Swipe Recognizer. https://developer.apple.com/documentation/uikit/uiswipegesturerecognizer

[8] Apple. 2018. Apple UI Guideline. https://developer.apple.com/design/human-interface-guidelines/

[9] Apple. 2018. Apple UIkit. https://developer.apple.com/documentation/uikit

[10] Apple. 2018. Touch Accommodations. https://support.apple.com/en-us/HT205269

[11] Cryptic Apps. 2018. Hopper - The macOS and Linux Disassembler. https://www.hopperapp.com/

[12] Kevin Wayne Arthur, Nada Matic, and Paul Ausbeck. 2008. Evaluating Touch Gestures for Scrolling on Notebook Computers. In *CHI '08*

Extended Abstracts on Human Factors in Computing Systems (CHI EA '08). ACM, New York, NY, USA, 2943–2948. https://doi.org/10.1145/1358628.1358788

[13] Jeff Avery, Mark Choi, Daniel Vogel, and Edward Lank. 2014. Pinch-to-zoom-plus: An Enhanced Pinch-to-zoom That Reduces Clutching and Panning. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14)*. ACM, New York, NY, USA, 595–604. https://doi.org/10.1145/2642918.2647352

[14] 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 (2013), 297–302.

[15] Leah Findlater, Jon E. Froehlich, Kays Fattal, Jacob O. Wobbrock, and Tanya Dastyar. 2013. Age-related Differences in Performance with Touchscreens Compared to Traditional Mouse Input. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 343–346.

[16] 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 23Nd Annual ACM Symposium on User Interface Software and Technology (UIST '10)*. ACM, New York, NY, USA, 153–162. https://doi.org/10.1145/1866029.1866055

[17] 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 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 6056–6061.

[18] Google. 2018. Google Gesture Design. https://material.io/design/interaction/gestures.html

[19] Google. 2018. Google Map. https://www.google.com/maps

[20] Google. 2018. Google Play Store. https://play.google.com/store

[21] Google. 2018. Touch and Hold Delay. https://support.google.com/accessibility/android/answer/6006989

[22] Tiago Guerreiro, Hugo Nicolau, Joaquim Jorge, and Daniel Gonçalves. 2010. Towards Accessible Touch Interfaces. In *Proceedings of the 12th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '10)*. ACM, New York, NY, USA, 19–26.

[23] Eve Hoggan, John Williamson, Antti Oulasvirta, Miguel Nacenta, Per Ola Kristensson, and Anu Lehtio. 2013. Multi-touch Rotation Gestures: Performance and Ergonomics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 3047–3050.

[24] 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 (2012), 1038–1043.

[25] Masatomo Kobayashi, Atsushi Hiyama, Takahiro Miura, Chieko Asakawa, Michitaka Hirose, and Tohru Ifukube. 2011. Elderly User Evaluation of Mobile Touchscreen Interactions. In *Human-Computer Interaction – INTERACT 2011 (INTERACT '11)*, Pedro Campos, Nicholas Graham, Joaquim Jorge, Nuno Nunes, Philippe Palanque, and Marco Winckler (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, Germany, 83–99.

[26] Heidi Koester, Rich Simpson, and Jennifer Mankowski. 2013. Software wizards to adjust keyboard and mouse settings for people with physical impairments. *J Spinal Cord Med* 36, 4 (jul 2013), 300–312.

[27] Heidi Horstmann Koester and Jennifer Mankowski. 2014. Automatic Adjustment of Mouse Settings to Improve Pointing Performance. *Assistive Technology* 26, 3 (2014), 119–128.

[28] Heidi Horstmann Koester and Jennifer Mankowski. 2015. Automatic Adjustment of Keyboard Settings Can Enhance Typing. *Assistive Technology* 27, 3 (2015), 136–146.

[29] Jhe-Wei Lin, Chiuan Wang, Yi Yao Huang, Kuan-Ting Chou, Hsuan-Yu Chen, Wei-Luan Tseng, and Mike Y. Chen. 2015. BackHand: Sensing Hand Gestures via Back of the Hand. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology (UIST '15)*. ACM, New York, NY, USA, 557–564. https://doi.org/10.1145/2807442.2807462

[30] Alexander Mertens, Nicole Jochems, Christopher M. Schlick, Daniel Dünnebacke, and Jan Henrik Dornberg. 2010. Design Pattern TRABING: Touchscreen-based Input Technique for People Affected by Intention Tremor. In *Proceedings of the 2Nd ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS '10)*. ACM, New York, NY, USA, 267–272.

[31] Kyle Montague, Vicki L. Hanson, and Andy Cobley. 2012. Designing for Individuals: Usable Touch-screen Interaction Through Shared User Models. In *Proceedings of the 14th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '12)*. ACM, New York, NY, USA, 151–158.

[32] Kyle Montague, Hugo Nicolau, and Vicki L. Hanson. 2014. Motor-impaired Touchscreen Interactions in the Wild. In *Proceedings of the 16th International ACM SIGACCESS Conference on Computers & Accessibility (ASSETS '14)*. ACM, New York, NY, USA, 123–130.

[33] 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 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 1934–1946.

[34] Maia Naftali and Leah Findlater. 2014. Accessibility in Context: Understanding the Truly Mobile Experience of Smartphone Users with Motor Impairments. In *Proceedings of the 16th International ACM SIGACCESS Conference on Computers & Accessibility (ASSETS '14)*. ACM, New York, NY, USA, 209–216.

[35] Alexander Ng and Stephen Brewster. 2017. An Evaluation of Touch and Pressure-Based Scrolling and Haptic Feedback for In-Car Touchscreens. In *Proceedings of the 9th International Conference on Automotive User Interfaces and Interactive Vehicular Applications (AutomotiveUI '17)*. ACM, New York, NY, USA, 11–20. https://doi.org/10.1145/3122986.3122997

[36] Francisco Nunes, Paula Alexandra Silva, João Cevada, Ana Correia Barros, and Luís Teixeira. 2016. User interface design guidelines for smartphone applications for people with Parkinson's disease. *Universal Access in the Information Society* 15, 4 (nov 2016), 659–679.

[37] NVIDIA. 2018. NVIDIA Cuda. https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html

[38] Sayan Sarcar, Jussi P.P. Jokinen, Antti Oulasvirta, Zhenxin Wang, Chaklam Silpasuwanchai, and Xiangshi Ren. 2018. Ability-Based Optimization of Touchscreen Interactions. *IEEE Pervasive Computing* 17, 1 (2018), 15–26.

[39] 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 (2012), 425–436.

[40] Shari Trewin. 2004. Automating Accessibility: The Dynamic Keyboard. In *Proceedings of the 6th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '04)*. ACM, New York, NY, USA, 71–78.

[41] Shari Trewin, Cal Swart, and Donna Pettick. 2013. Physical Accessibility of Touchscreen Smartphones. In *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '13)*. ACM, New York, NY, USA, 19:1–19:8.

[42] Huawei Tu, Xiangshi Ren, Feng Tian, and Feng Wang. 2014. Evaluation of Flick and Ring Scrolling on Touch-Based Smartphones. *International Journal of HumanâĂŞComputer Interaction* 30, 8 (2014), 643–653. https://doi.org/10.1080/10447318.2014.907017 arXiv:https://doi.org/10.1080/10447318.2014.907017

[43] 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 SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 623–626.

[44] Karl Wiegand. 2015. Impact of Motor Impairment on Full-Screen Touch Interaction. *Journal on Technology and Persons with Disabilities* 3, 22 (2015), 58–76.

[45] Jian Zhao, R. William Soukoreff, and Ravin Balakrishnan. 2011. A model of multi-touch manipulation. In *Proceedings of the 2nd Annual Meeting of the Graphics, Animation, and New Media (GRAND '11)*. GRAND NCE, Ottawa, Ontario, Canada, 58–76.

[46] Yu Zhong, Astrid Weber, Casey Burkhardt, Phil Weaver, and Jeffrey P. Bigham. 2015. Enhancing Android Accessibility for Users with Hand Tremor by Reducing Fine Pointing and Steady Tapping. In *Proceedings of the 12th Web for All Conference (W4A '15)*. ACM, New York, NY, USA, 29:1–29:10.