EVALUATING FINGER ORIENTATION FOR POSITION AWARENESS ON MULTI-TOUCH TABLETOP SYSTEMS

by

HONG ZHANG

A thesis submitted to the Faculty of Graduate Studies of
the University of Manitoba
in partial fulfilment of the requirements of the degree of

MASTER OF SCIENCE

Department of Computer Science
University of Manitoba
Winnipeg, Manitoba, Canada

Copyright © 2012 Hong Zhang

ABSTRACT

Interactive tabletop systems are becoming popular platforms for group activities. However, current common tabletops do not provide capabilities to differentiate interactions among simultaneous users, i.e. to associate a touch point with its proper owner. My thesis proposes and explores the use of an important biometric property of users as the basis for touch discrimination on multi-user tabletops: Finger Orientation (FO).

In this thesis, I first collect the FO ranges of users standing in different positions around a tabletop. Second, I implement a system that uses FO to determine where the users are standing, and based on that extrapolate the owner of the touch. Next, I evaluate the system with two separate experiments, present the results, and discuss all findings. Furthermore, I explore some enhancements with a simple quantitative study. My results indicate that finger orientation is a good natural biometric trait enhances multi-user recognition on tabletops.

PUBLICATIONS

Some ideas and figures in this thesis have appeared previously in the following publications by the author:

Hong Zhang, Xing-Dong Yang, Barrett Ens, Hai-Ning Liang, Pierre Boulanger, Pourang Irani. See Me, See You: A Lightweight Method for Discriminating User Touches on Tabletop Displays In *CHI 2012*, May 5-10, 2012, Austin, TX, USA.

The following figures have been reproduced from other sources with permission:

Figure 2a and 2b on page 5 from [26] © 2008 authors.

Figure 3 on page 8 from [13] © 2011 authors.

Figure 4 on page 9 from [7] © 2001 ACM, Inc.

Figure 5a on page 11 from [16] © 2010 ACM, Inc.

Figure 5b on page 11 from [15] © 2010 ACM, Inc.

Figure 6a and 6b on page 12 from [11] © 2010 ACM, Inc.

Figure 8 on page 12 from [4] © 2011 ACM, Inc.

Figure 14 on page 27 from [12] © 2009 authors.

Life is not a problem to be solved,
but a reality to be experienced.

— Soren Kierkegaard

ACKNOWLEDGMENTS

First and foremost, I would like to thank Dr. Pourang Irani for his incredible personal and professional support. He is a great advisor and mentor. Thank you Pourang.

I thank my family for their support and confidence in me, especially to my wife Jing for giving me a beautiful daughter Olivia Xing-Ru Zhang in 2011.

Thank you to my friends. You are all amazing – never once doubting that I should be doing this or that I was capable. You all provided an outlet for frustrations, an ear for my half-brained ideas, tried out my experiments and read my work. Thank you.

Thank you to my collaborators: Dr. Pourang Irani, Dr. Hai-Ning Liang, Dr. Xing-Dong Yang, and Barrett Ens. I learned so much working with you.

Thank you to my committee members, Dr. Ekram Hossain and Dr. Rasit Eskicioglu, for your time, helpful comments, and support.

Thank you again to Dr. Irani, also to Faculty of Graduate Studies, Faculty of Science, Graduate Students' Association, for their generous financial support throughout my degree.

And finally, thank you to my fellow students in the HCI lab, in particular Cary Williams for helping to setup the touch table and troubleshoot all hardware problems, but also to Khalad Hasan, David McCallum, and Hina Aman for your support.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

ANOVA	Analysis of Variance
CCV	Community Core Vision
DI	Diffuse Illumination
DSI	Diffuse Surface Illumination
FO	Finger Orientation
FTIR	Frustrated Total Internal Reflection

HCI	Human-Computer Interaction
IA	Identity Awareness
PA	Position Awareness, Proximity Awareness
PAC	Position Awareness Cursor
SMESY	SEE ME SEE YOU
SVM	Support Vector Machine

1

INTRODUCTION

Multi-touch tabletop systems (tabletops for short) provide a shared environment for users to work together. To best support multi-user tasks, tabletops should be able to differentiate the actions of one user from those of another. Unfortunately, this feature is missing in current common tabletop systems. I refer to this limitation as ***touch indiscrimination***, which restricts the potential extension of multi-user tabletop applications. In a game application, for example, responsibility falls on individuals for moving the correct pieces or taking their turn at the right time. Awkward solutions must be found for discriminating touches in a painting program, such as defining explicit user territories [31, 32], or requiring gestures to delineate every input [22].

Because tabletop systems are inherently collaborative, solutions have been explored to make them **touch-discriminate**. The feature of touch discrimination enables application designers to support interactions that would not be otherwise possible. Figure 1 demonstrates three users using a drawing application on a touch table. Each of them can draw with a different color and a different pen thickness without interfering with others.

Some investigations exist to solve the touch indiscrimination limitation. One approach is to use an identifying device, held or worn

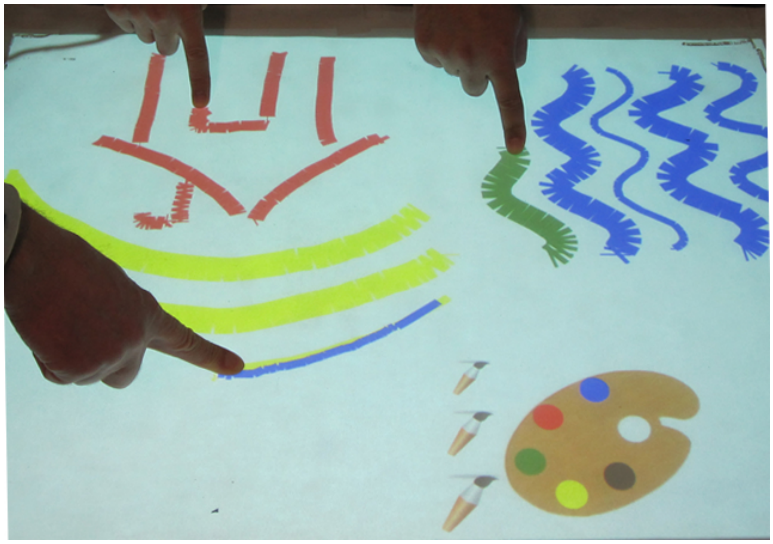


Figure 1: Users interacting with a drawing application with one shared color palette using touch-discriminate technique: SEE YOU SEE ME (described later in this thesis). This form of collaborative work would not be possible without maintaining distinct user states.

by the user, as a proxy for the actual owner of a touch point [7, 27]. Unfortunately this approach requires extensive modification to the tabletop system and the use of peripheral accessories. Another approach is to employ both hands of a user, such as IdLenses [31]. However, this approach is either difficult for users to adapt to or lack proper multi-user evaluations.

I have identified a number of problems with the existing techniques for discriminating users' touch point and proposed another approach for identifying touches from different users. I define the term **Position Awareness** (PA) to a system feature that associates a touch with a user's position and I design a new technique called SEE ME SEE YOU that employs users' finger orientation to achieve PA touch. SEE ME SEE YOU allows the design of applications that differentiate user's touch. The rest of this thesis is structured as follows: In Chapter 2, I discuss the work related to touch table and

touch discrimination. I then present, in Chapter 3, a detailed problem statement, a list of design criteria of SEE ME SEE YOU, and the implementation to extract finger orientation. Followed, in Chapter 4, with a complete evaluation of the system. In Chapter 5, I present two enhancements of SEE ME SEE YOU and discuss a qualitative study to collect users' impression of the system. Finally, in Chapter 6, I provide a summary of the thesis and some directions for future work.

2

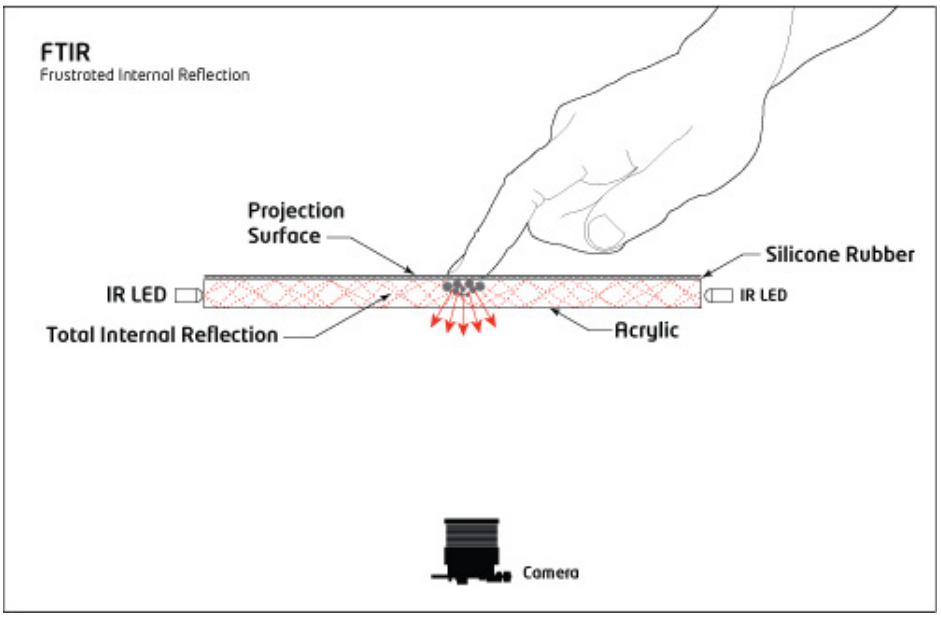
BACKGROUND AND RELATED WORK

Researchers have studied tabletop systems for many years. In this chapter, I first go through implementations of current popular tabletops. Second I survey existing methods to differentiate users' touches. Following this, I introduce researches that lead to my touch discrimination technique.

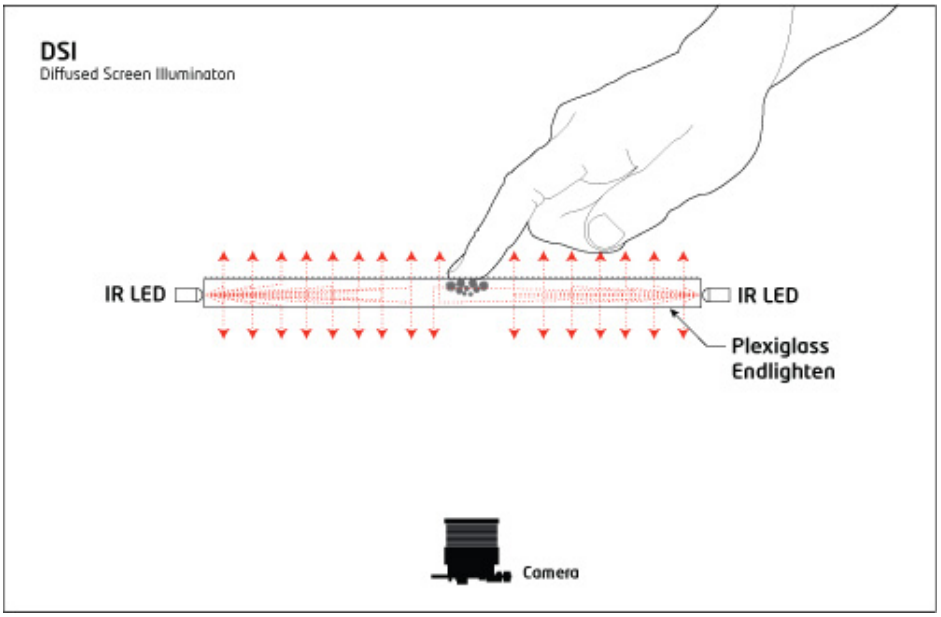
2.1 INTERACTIVE TABLETOPS

The development of interactive tabletops has gained significant interest in recent years. Newman and Wellner [19] show their pioneering work — the Digital Desk, which consists of a physical desk with a video camera and a projector installed above. The projector projects electronic images down onto the desk, and the camera tracks the movements of an LED-tipped pen so that the user can use the pen to interact with the computer. Later, Fitzmaurice et al. [9] develop the Active Desk, a large desk with a rear-projection computer screen underneath the surface. Next, Dietz and Leigh [7] introduce DiamondTouch, which uses a capacitance based touch surface. Their surface transmits electrical signals from the touch point through the user to a receiver connecting with a computer. Sensetable, presented by Patten et al. [23], rely on a different technology. They use electro-

magnetics under the table to track the positions and orientations of multiple wireless objects on the surface. Similarly, SurfaceFusion [20] used Radio Frequency Identification (RFID) technology to determine the location of tagged objects.



(a) FTIR (Frustrated Total Internal Reflection)



(b) DSI (Diffuse Surface Illumination)

Figure 2: Vision-based tabletop implementation (a) FTIR; (b) DSI. Images from [26].

Today, one popular approach that has been widely used to implement a multi-touchable tabletop is vision-based system. Vision-based approaches typically involve image processing to determine interactions on the surface. Han [10], in 2005, showed a low-cost tabletop implementation through Frustrated Total Internal Reflection (FTIR) principle (figure 2a). He injects strips of LEDs around a transparent acrylic panel's edge and places an infrared-sensitive camera perpendicular to the surface. When a user touches the surface, the light is reflected by the finger and caught by the camera. The camera images will be applied with a basic set of computer vision algorithms to determine the contact point. The FTIR technology works well for detecting touches. However, it struggles with dragging operations because the light reflection requires a certain pressure, but when users drag, they will apply less pressure on the surface, which causes that the tabletop could lose track during dragging.

A similar vision-based technique is the Diffuse Illumination (DI) system [26]. Instead of injecting the infrared lighting to the surface, DI places it behind the projection surface. One drawback of the DI system is infrared lighting can spread unevenly across the screen surface. An improved version called Diffuse Surface Illumination (DSI) [26](figure 2b) reduces this drawback. DI and DSI both suffer the problem of false touch detection because of their high sensitivity, but they are able to provide features such as detecting objects and in-air operations.

Andreas [12] attempts to combine both FTIR and DI in one tabletop. He uses two sets of infrared lights: one for FTIR and another for DI. He also installs a special designed circuit to switch on/off different

lights to catch one FTIR raw image and one DI raw image. He shows that his tabletop can combine advantages from FTIR and DI to not only improve touch detection but also offer more powerful detection, including left or right hand, fingertip, and hand orientation detection. A vision-based multi-touch tabletop system can be easily built and has been widely used by many researchers due to the low cost of the required hardware and the availability of the supported software [1].

In addition to those vision-based methods, there are some other technologies that can be used to build a multi-touch table. Capacitive Touch Technology ([18, 25, 35]) overlays a capacitive touch screen on top of an existing screen to turn it into a multitouch display; however, this approach is not only expensive but also only supports a limited number of simultaneous touches. Digital Vision Touch (DViT) invented by Smart Technologies [34] makes use of the cameras that are installed in each corner of the screen. Multiple cameras communicate with each other through their sophisticated software to determine the position of the touch. PixelSense [13] from Microsoft does not rely on optical computing, instead, their technology shoots infrad lights from the bottom layer of the screen. The lights are then reflected by the touches and captured by the integrated sensor (see Figure 3). A DViT smart table is costly and Microsoft Surface 2 is not yet on the market yet (at the moment this thesis was written, Microsoft had started accepting pre-orders).

Tabletops are popular because of their support of multiple users. When Dietz and Leigh [7] develop DiamondTouch, they indicate that they aim to address the problem of using multiple mice in a collaborative environment. They describe a situation when several

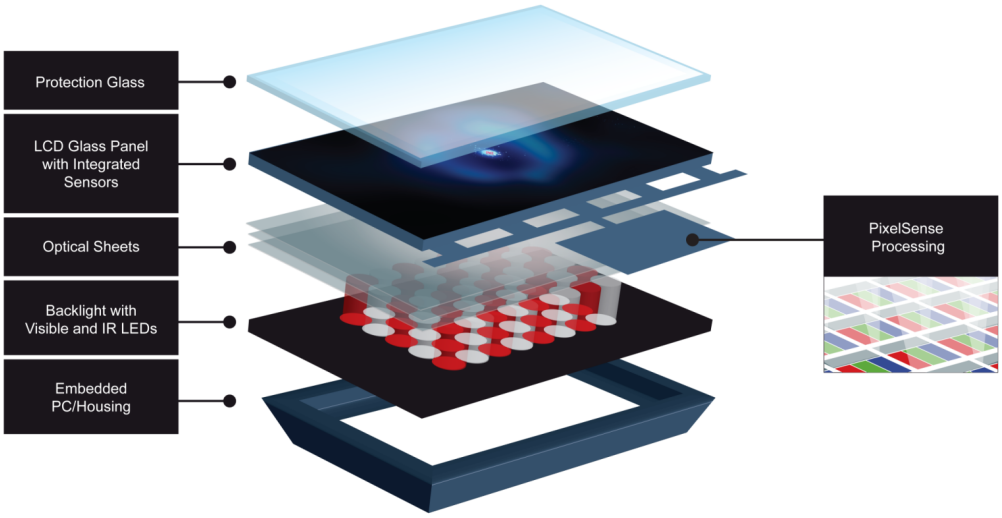


Figure 3: Layers of internals needed for PixelSense to work. (Image from [13] © Microsoft. Reprinted by permission.)

users work on one screen and because each user is using a different mouse, the multiple on-screen cursors is visually confusing and make distinguishing one cursor from the other difficult. Moreover, given mouse-based operations do not easily transfer to a tabletop, direct touching is initially used to replace the mouse to support multiple users working together in one computer system. However, unlike the mouse, direct touch does not have a cursor associated with a specific user so for the computer system to identify what operation is performed by which user.

2.2 TOUCH DISCRIMINATION ON TABLETOPS

Touch Discrimination is a feature that can identify who is interacting with the tabletop. Several researchers([21, 28, 29]) have pointed out the benefits of having touch discrimination on tabletops. For

example, multiple users can select different tools from a shared toolbox without interfering with each other; the application is able to log each user's behaviour; and each user is able to revert his or her most recent action(s). In addition, the application can respond differently according to the current interacting user.

2.2.1 *Fixed Location*

Partridge [21] refers identity awareness (IA) to be a system feature that associates each operation with a particular actor. DiamondTouch [7] is one of the earliest tabletops known for its IA capabilities. Figure 4 depicts how DiamondTouch works. It employs a pad placed beneath a user's seat to create a close circuit between each user and the tabletop. As such, it identifies users based on their seat positions.

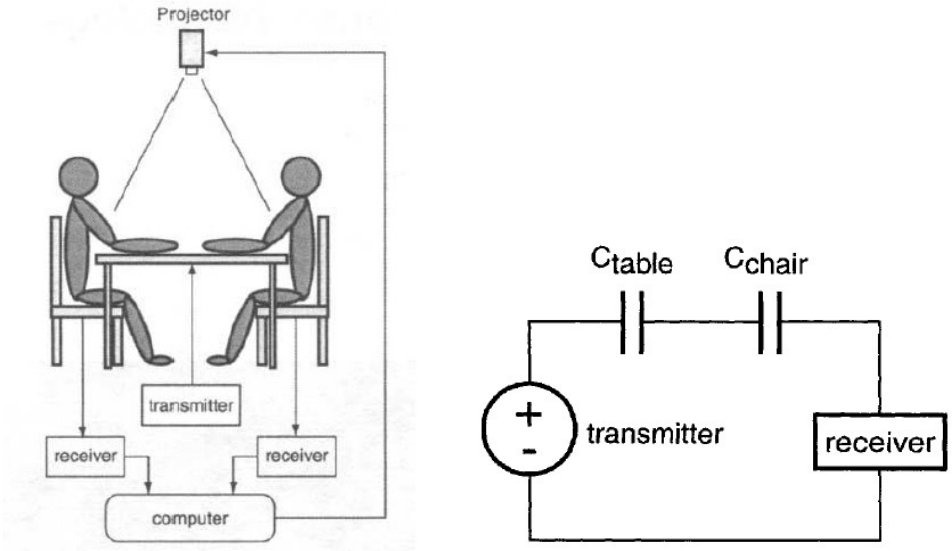


Figure 4: Configuration of DiamondTouch [7]: when a user touches the table, the table transmits electrical signals from the touching point through the user to the receiver (chair) connecting with a computer. (Image from [7] © 2001 ACM, Inc. Reprinted by permission.)

The IA capabilities of the DiamondTouch are instrumental for several projects, including: the UbiTable [33], where users are given their own documents for annotation; SIDES [24], a game designed to help children with Asperger's syndrome to improve their social skills; and TeamTag [17], a system that supports multiple users' exploration and annotation of digital photos. DiamondTouch provides a reliable hardware solution, but does not work with common vision-based systems. It operates best when users are seated and in a fixed position around the tabletop.

2.2.2 *Wearable Devices*

Another common strategy for IA systems involves the use of some external device that the system can easily recognize. Myer and Schmidt [16] use uniquely identified wristbands (IdWristbands), equipped with infrared LEDs that transmit coded light pulses, as identifiers (Figure 5a). This approach works with vision-based touch tables. The table's IR camera is able to catch the LED signals emitted from the wristband and allow the system to associate the touch with a specific user. Similarly, Roth et al. [27] present the IR Ring, a ring-like device that emits a distinguishable light signal, around a user's finger.

Marquardt et al. [15] introduce fiduciary-tagged gloves to distinguish one user's hand from others (Figure 5b). They use 2×2 cm fiduciaries and glue them to a regular glove. Each tag carries information including: an 8-bit identification, the coordinate when a touch happens, and the orientation to the table. Their solution is not

Figure 5: (a) IdWristbands (Image from [16] © 2010 ACM, Inc. Reprinted by permission.) (b) Fiduciary-tagged Gloves (Image from [15] © 2010 ACM, Inc. Reprinted by permission.)

only inexpensive but also able to identify which part of the user's hand is interacting with the table.

The use of wearable devices reduces the hardware requirements of a system such as DiamondTouch and also removes the constraint, such as a user must be seated in a fixed location. However, these systems share a main drawback which requires users to wear an accessory. Using additional accessories limits the use of these systems in certain contexts. In public settings, for example, the attachments can potentially get lost, misplaced or may be unfit for some users. Moreover, some users might refuse to use the accessory because of personal hygiene.

2.2.3 *Using Biometric Traits*

It is possible for a tabletop to provide user discrimination without users needing to wear accessories, but relying on biometric traits. Fingerprints [11] is studied by Holz and Baudisch, as fingerprints are distinct from one person to another. However, the goal of their study is not intended to differentiate touch points and also not done

on a tabletop. They tend to use the extracted fingerprints to improve touch accuracy.

(a) Holz and Baudisch's [11] show fingerprint outline and features move in synchrony when dragging but remain stationary when rolling.

(b) RidgePad prototype

Figure 6: The study of using fingerprints to improve touch accuracy. (Image from [11] © 2010 ACM, Inc. Reprinted by permission.)

In their study, Holz and Baudisch [11] show an interesting observation from their study that fingerprint outline moves when users are interacting with the system (Figure 6a). They present algorithms that accurately keep track of touches using fingerprints and demonstrate a prototype device called RidgePad. Unfortunately, to my knowledge, today there is no fingerprint scanner that can be easily integrated with tabletops to provide touch discrimination. When sophisticated fingerprint scanning becomes available and affordable, fingerprints can be used as a promising biometric trait to discriminate user touches. Until then, other solutions are necessary.

Schmidt et al. [30] explore the contours of users' open palms to identify users. They also present a technique called IdLenses [31], which allows users to use their non-dominate hand to trigger a virtual lens. The lens represents the user's personal space (Figure 7). Every action in the registered lens is considered to belong to the user who triggers the lens. Their approach can be easily applied to any

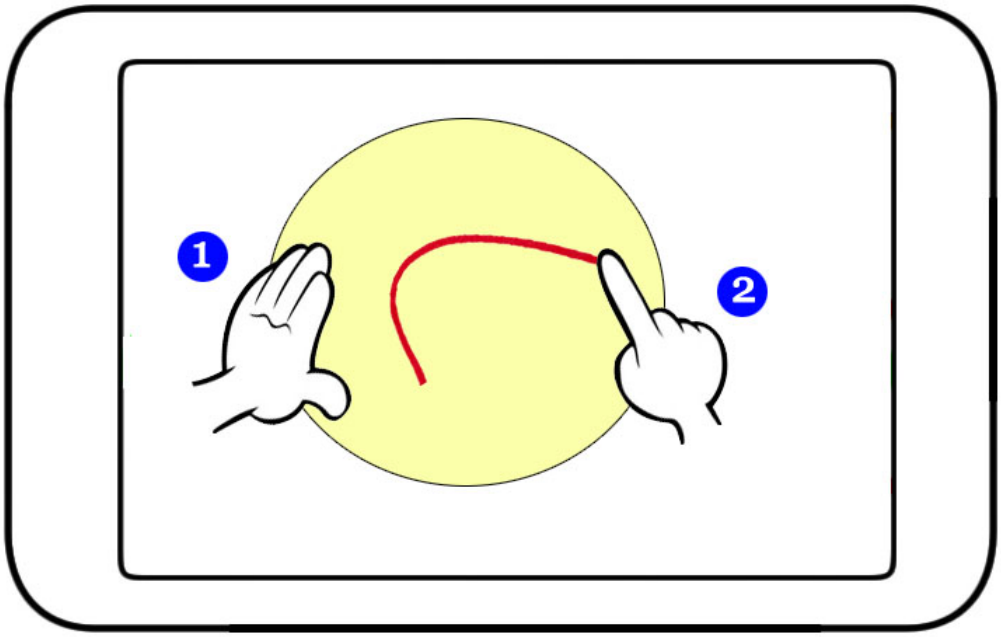


Figure 7: IdLenses [31] requires two steps: (1) the user places down the hand to register a lens on the table; (2) the user operates inside the registered lens and those operations are identified (this figure is a remake from my own interpretation).

tabletop system, including vision-based tabletops. The flip side of their technique is that users have to employ both hands to complete even a simple IA operation like tapping.

Dohse et al. [8] identify a user's location by tracking the user's hands. They augment an FTIR tabletop with an overhead camera and use a skin color detection algorithm to track different users' hands. However, they did not conduct any usability test with their system. Because their system has to communicate with an additional overhead camera, the efficiency and the accuracy of their system remain unknown.

Dang et al. [6] develop a heuristic method based on the positions and angles between multiple fingers to determine if a touch belongs

to the left or right hand and extrapolate the position of its owner. However, although they distinguish which hand or even which finger is touching, they mainly focus on using finger orientation to enhance gesture recognition and touch interaction on tabletops, and have not explored associating touches with users.

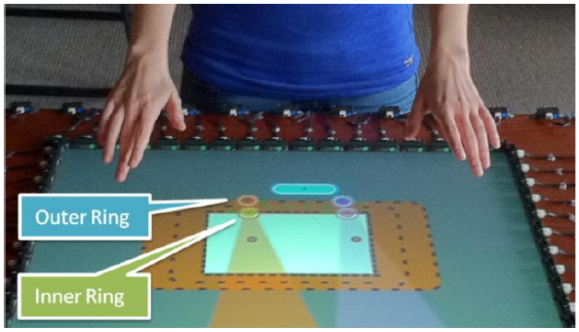
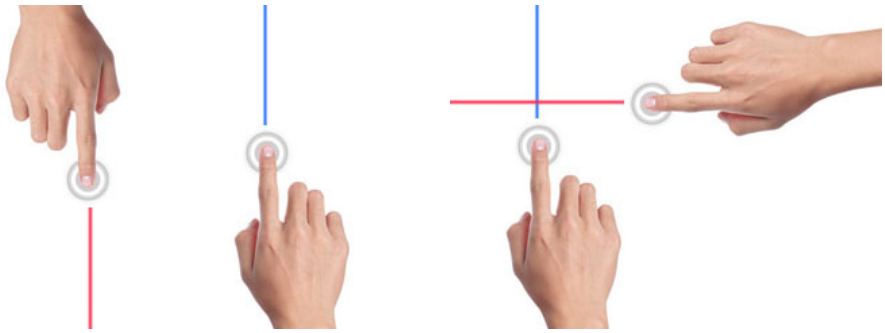


Figure 8: Medusa [4] visualize the user's body using a blue paddle, the right arm with an orange circle, and the left arm with a purple circle. The system projects the user's arms with an orange and a purple cone (for right and left arm, respectively). (Image from [4] © 2011 ACM, Inc. Reprinted by permission.)

Finally, Annett et al. [4] present Medusa, a tabletop that can sense users' position (see Figure 8). They augment a Microsoft Surface multi-touch table with 138 inexpensive IR-based proximity sensors that are installed to the top and the side of the table. In addition, they implement a set of unique user interactions within their table. Despite the inspiring idea of user position sensing and the novelty of their new interaction, they fail to associate a touch point with its proper owner due to the limitation of the proximity sensors.

2.3 USERS' FINGER ORIENTATION

To explore a possible approach to achieve touch discrimination, I have studied the use of Finger Orientation (FO). FO is referred to the direction of the touch point on the table. Theoretically, FO is useful for IA on tabletops because a user's FO profile can be unique according to where the user is standing. For example, when two users are standing face to face around the table (Figure 9a), their FO profile should be very distinct. The difference of the fingertip pointing direction at the same touching point from these two users should be nearly 180°. Similarly, users' FO can be also distinguishable when two users are standing perpendicularly (Figure 9b) since their profile will have nearly 90° difference. However, it is not quite clear whether FO will be different enough for users standing side by side.



(a) When two users are face to face, their FO profile should differ 180°.

(b) When two users are perpendicular, their FO profile should differ 90°.

Figure 9: Theory of using finger orientation to differentiate user touches. The blue and red line indicate the user's finger pointing direction.

Wang et al. [37] present an algorithm to collect FO data from vision-based tabletops. Their algorithm relies data from a finger landing in an oblique method (where the fingertip is placed first, followed by a roll to the full finger pad) to extract the major axis of a touch blob

(see figure 10) given by the table's computer vision software. At each input frame the algorithm conducts a connected component analysis to extract all finger contact regions. For each contact point, four steps take place. First the algorithm fits the contact shape into its most perfect elliptical shape. Second, it assesses whether the finger is in an oblique touch state, based on the shape's overall area and its aspect ratio. Third, the algorithm considers the finger's landing dynamics to determine in which direction the finger is pointing. Finally, when the finger's blob stabilizes, detection ends and final orientation is reported.

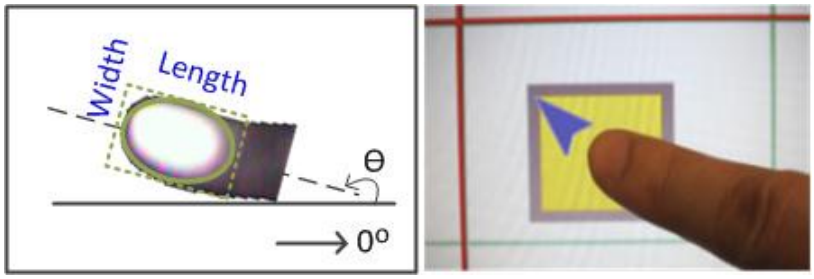


Figure 10: Left: finger blob captured by tabletop's IR camera upon an oblique landing and used to extract the finger orientation (Right).

Wang et al.'s [37] algorithm requires the users to touch in a motion they call an 'oblique touch', where the fingertip is placed first, followed by a roll to the finger pad. This allows extraction of both the orientation and direction of the major axis of an ellipse defined by a touch blob. In a lab of implementation of their algorithm, a series of proof-of-concept studies show that the unintuitive nature of this oblique landing constraint make FO detection unreliable without extensive user training. Roughly 20% of trials resulted in a finger orientation inverted by 180°.

Dang and Andre [5] present another algorithm. They first extract user hand contours from the camera's raw images, next determine the symmetric lines that wrap the finger contour, and finally average the tracked points on the lines till they can derive the finger angle. They show that their algorithm can achieve 94.87% recognition rate when the error tolerance is $+\backslash-$ 10 degree. Their work is primarily concerned with the evaluation of their algorithm and compare the accuracy to other methods, such as naive ellipse method. In contrast, although based on a similar approach that uses hand contour, I implement a different algorithm to extract FO. Details will be described in the next chapter.

3

SYSTEM DESIGN

Based on previous research, although interactive techniques on multi-touch tabletops have been actively studied, only a few have tried to tackle the challenges of distinguishing simultaneous touches from multiple users. In this chapter, I describe the requirements of a lightweight touch discrimination technique. I next classify finger orientation as a lightweight technique. In the end, I present a simple algorithm to extract finger orientation from a vision-based tabletop.

3.1 LIGHTWEIGHT TECHNIQUE

Touch discrimination on tabletops is still an open problem. My primary research interest is to develop a lightweight technique to achieve user touch discrimination. A lightweight technique shall allow users ease of use and adoption. In addition, the technique shall enable designers to integrate it into existing systems with minimal effort. Finally, all required software and hardware shall be inexpensive and easy to acquire. Therefore, a lightweight touch discrimination technique should meet the following criteria:

1. **Minimal device constraints:** the system should not require users to hold or wear an external device;

2. **Accurate:** the system should be accurate enough to not overburden or distract users from their primary tasks;

3. **Scalable:** the system should be versatile enough to handle various configurations such as multiple simultaneous users, users standing side-by-side, and uniform accuracy coverage across different regions;

4. **Low cost:** building the tabletop should be achievable at an affordable cost with commonly available technology; and

5. **Computationally non-prohibitive:** the system should work in real-time and not suffer from excessive lag.

To facilitate the engineering of a lightweight technique, I restrict my expectations with some additional caveats:

1. **Limited input features:** users benefiting from a lightweight system may be willing to forgo certain types of multi-touch use, such as using the full palm to interact with objects. This would allow them to make the best use of the device's touch discriminating features;

2. **Implicit trust:** the system should be designed for users who intentionally want touch discrimination. A lightweight system need not prevent identity deception as this would add layers of complication to normal use;

3. **User adaptation:** although a lightweight system should not require long training periods, some knowledge about how the system operates can contribute to improved usage and a better user experience.

3.2 DESIGN OF SEE ME SEE YOU

To design my vision of a lightweight technique, I first define the term Position Awareness (PA) as a system feature that associate an operation with a user's standing position. The definition of PA combines Partridge's [21] definition of IA and User Proximity Sensing from Annett et al. [4]. In a multi-user tabletop environment, users stay relatively stationary for an interaction task or set of scenarios. Therefore, I can safely associate an input with a specific user. Once the user has the need to change position, other technologies (will be discussed in section 6.1) can be introduced to trace the user's movement. I consider that to be beyond the scope of my research.

Next, I choose to use finger orientation in my system for four reasons: 1) as discussed in section 2.2.3, in theory, users' FO profile can be distinct according to where the user is standing; 2) I can use FO values to obtain information about users' standing position to enable PA detection; 3) using FO directly requires minimal training to users and no additional device, hence it matches the design criteria of a lightweight technique; 4) although some researchers([6, 12, 37]) have studied finger orientation, no one has done a complete study to evaluate the feasibility of using it for touch discrimination.

Furthermore, I choose a vision-based FTIR multi-touch table over other technologies because vision-base tabletops are low-cost. In addition, a vision-based tabletop allows me to perform optical processing to extract FO easily. But one potential benefit of using FTIR over DI is that an FTIR tabletop can avoid early touch detection which can occur with DI systems. The main goal of my thesis is to

evaluate whether FO is appropriate to use for touch discrimination on tabletops. Therefore, the choice of the tabletop is less relevant to the study if other systems can easily read the FO values in the future.

The lightweight system SEE ME SEE YOU (SMESY), is conceived to be a quick and easy method which differentiates user touches on multi-touch tabletops. Although SMESY depends on accurate determination of FO, the benefits of the technique are independent from any particular FO detection algorithm. Once FO is accurately assessed, I associate user touches with user positions using a machine learning algorithm. I choose this method over a heuristic approach for ease of implementation and robustness due to the ability of such algorithms to generalize given limited training data. Although I choose a Support Vector Machine (SVM) classifier for touch association, the system may be implemented using any adequate classifier of the developer's choosing.

3.3 MY FINGER ORIENTATION ALGORITHM

Below I describe my FO algorithm designed specifically for vision-based tabletop systems.

3.3.1 *Table Implementation*

I use a custom-built FTIR [10] tabletop (similar to Figure 2a) with dimensions of 26 in. (length) $\times$ 20 in. (width) $\times$ 36 in. (height) (Figure

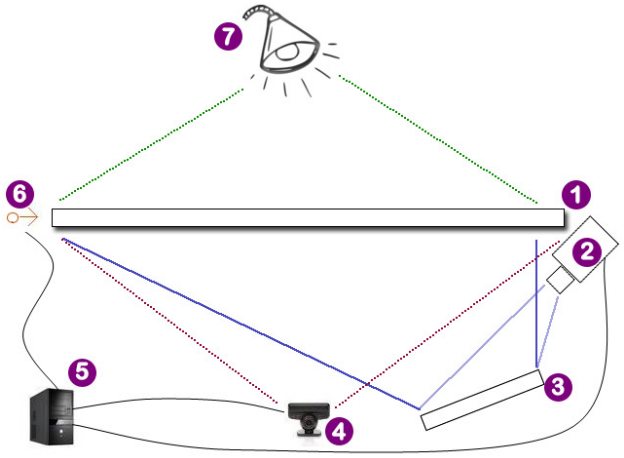


Figure 11: Hardware Configuration of SMESY.
(1) Touch Surface (acrylic glass sheet);
(2) Projector;
(3) Mirror;
(4) PS3 Camera;
(5) Computer;
(6) IR Light Strip for FTIR; and
(7) Overhead Lamp.

11). The tabletop uses infrared LED lamps (Figure 11.6) emitting light with a wavelength of 850 nm using a 12 volt power supply and a Vivitek Qumi projector (Figure 11.2) with a 1280 × 800 resolution and a brightness of 300 lumens. Because the table is not high enough, I use a mirror (Figure 11.3) to reflect the projection to the surface (Figure 11.1) so that the computer image can be fully displayed. The experimental platform uses the TUIO protocol with the Community Core Vision (CCV) tracker [1], and runs on a 1.86 GHz Core 2 Duo PC with Windows XP (Figure 11.5). To cycle the LEDs for hand contour extraction, I use a Phidgets 3052 SSR relay board [2]. The table's built-in IR camera (Figure 11.4) captures a 640 × 480 image at a rate of 60 fps. Due to cycling the camera frames for alternate use by the CCV server and for hand contour analysis, the resulting frame rate is 20 fps.

My algorithm relies on hand contours, which can be obtained with a standard DI setup, or with FTIR, given the following modifications. To obtain clear and complete hand contours for my evaluation, I placed an overhead lamp (Figure 11.7) above the FTIR table. To reduce obfuscation caused by the imbedded infrared light array, a relay controller is introduced into the IR lighting circuit to cycle the lights on and off. In this way I capture a precise hand silhouette image (Figure 12a) for each cycle of the FTIR vision server.

3.3.2 *Algorithm Details*

The raw image is cropped around the coordinates of touch blobs that are detected by CCV to extract the contour of the touching hand (Figure 12a, inset). I then derive the direction of the pointing finger from the hand contour image by examining a circular slice of pixels that lie within a radial range from the center of the touch blob (Figure 12c). The inner circle in yellow (Figure 12b) represents the touch blob. My algorithm draws the outer circle, whose radius is equal to 8 pixel plus the inner circle's radius. Once I find the slice, I remove everything else in the image. According to the testings, the size of the 8-pixel radius works for both male and female hands. The remaining contour is the one in red. A line from the center of the remaining contour (in red) to the center of the touch blob (red line in Figure 12c) determines the FO angle.

Similar to Dang and Andre's [5] algorithm, I also use hand contour. Their algorithm is more generalized to support multiple fingers; however, their prediction rate does not seem high and they do not

test their results in realtime. Because my goal is to evaluate if users' finger orientation is feasible to use for touch discrimination, I develop a different, simpler, yet accurate algorithm. Moreover, since finger orientations differ from one finger to another, I choose to restrict my exploration to the index finger. Although my algorithm can be modified to detect the orientation of other fingers, I feel that this restriction is not detrimental because research [14] has shown most users extensively use index fingers on tabletop.

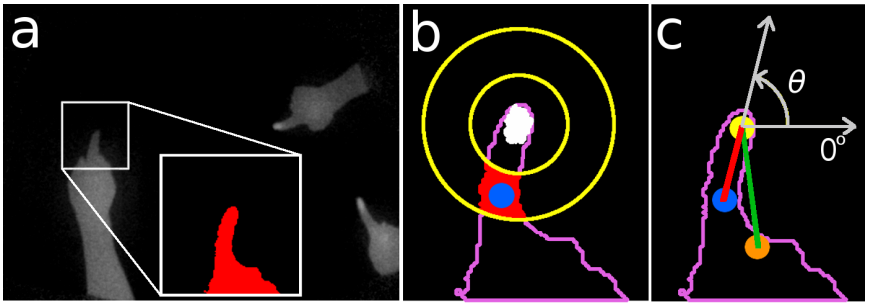


Figure 12: A silhouette of users' hands (a) is cropped and processed to find the contour of a touching hand. The contour is masked to reveal the area betIen two radii (b) around the FTIR touch blob received from the FTIR server. The finger orientation is given by a line (shown in red) from the touch blob to the center of the remaining area (c). A second line (in green) to the center of the hand contour determines if it is a left or right hand.

3.3.3 *Detecting Handedness*

In addition to detecting FO, my algorithm can be adapted to detect the handedness of user touches. When the line for finger orientation is determined, a second line is derived from the hand contour extraction (green line in Figure 12c). In this case, it is from the touch blob to the centroid of all pixels in the extracted hand mass. Assuming that the user is pointing with their index finger, I can determine handed-

ness with relatively high accuracy (>90%) by checking whether this second line lies to the left or right of the first line.

Figure 13 demonstrates some samples of the raw shadow image and the processed hand contour image. The red image shows the extracted hand contour from the raw image. The green line connects the touch point to the centroid of whole extracted hand. The white line connects the touch point to the centroid of the radial pixels I calculated. If the green line is on the right side of the white line, I consider it as right-hand touch. If the green line is on the left side of the white line, I consider it as a left-hand touch. Note that the raw images captured by the camera are reversed. The samples look like coming from the opposite hand, but those samples are correctly labelled.

Andreas [12] presents an approach to detect left or right hand by checking the difference of two distances: d1, the distance from the pinky finger to the ring finger; d2, the distance from the thumb to the index finger. If d1 is smaller than d2, the hand is detected as a left hand (see figure 14). His method requires the hand is fully open and he does not evaluate the accuracy of his approach.

In contrast, Benjamin et al. [36] implement a classifier using a decision tree algorithm to detect handness with the input of blob size, blob positions and arm orientation. Although their method, different than mine, considers more scenarios other than just the index-finger touches, they only obtained 80% accuracy. In my two studies (described in the next chapter), I achieve an accuracy of 91.26% and 92.44% respectively. To support other scenarios, such as

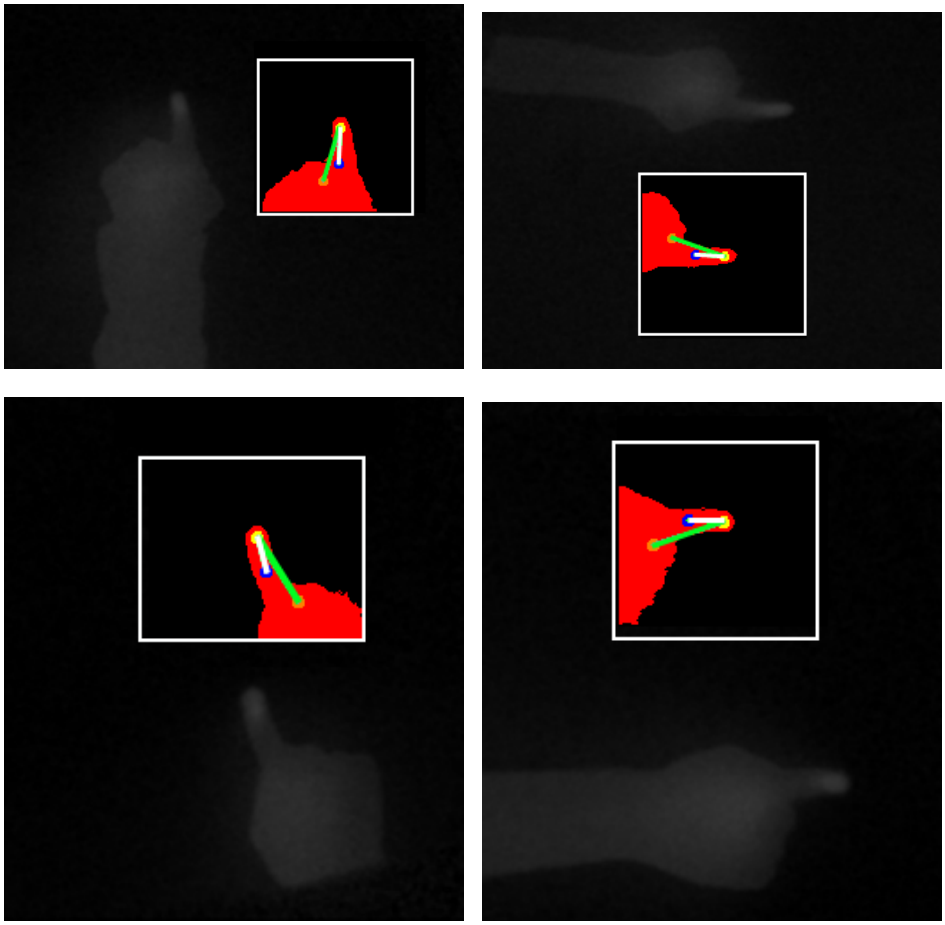


Figure 13: Two samples of right-hand touches (Top) and left-hand touches (Bottom). The grey image is the shadow of the user's hand. The image in the white box shows the hand contour and the indicators after processing.

open-hand touches is also possible. More details will be described in chapter 4.

3.4 PSEUDO CODE

This section includes a few key functions that are used in SMESY. SMESY has two parts: one is the vision server that modified from an open source project CCV [1], which is written in C++ with open-

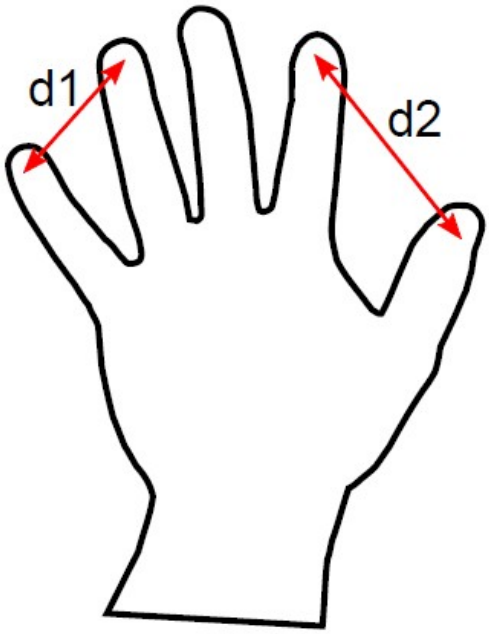


Figure 14: Andreas' [12] approach to detect handedness: if d1 is smaller than d2, the hand is a left hand. (Image from [12] © authors. Reprinted by permission.)

Frameworks library [3]; another is the front-end user interface (such as the experiment and the demo applications), which is written in C# with .NET framework 4.0.

3.4.1 *Using Phidget Controller*

The Phidget [2] control board is used to turn on and off the IR lights programmatically. To obtain the best result of the raw image, I apply a different gain value and exposure value to the PS3 camera after turning off the IR lights. They are restored after I take the frame I need.

```
function GetRawImages() : PBYTES
{
    TurnOffFTIR();
    ChangeGainAndExposure();

    // skip one frame for activating the setting
    Sleep(17);

    CLEyeCameraGetFrame(_cam, gainImageBuffer);
    TurnonFTIR();

    RestoreGainAndExposure();
    Sleep(17);

    return gainImageBuffer;
}
```

3.4.2 *Extract the Hand Contour*

To extract the hand contour, I firstly crop a smaller image from the raw image. The centroid of the small image is the touch point. According to my testing, cropping the image with a size of 120×120 pixels appears to serve a good result, since the image contains the whole hand and the image size is not too big to decrease the system performance. Secondly, Two input thresholds are customizable from the modified CCV [1] interface. But the processing image has been already grayscaled, so it has only one color channel. The low_threshold and the high_threshold specify the pixel values that likely belong to a hand. I use the openFrameworks library to find all possible contours from the processed image. If there is more than one contour, a simple algorithm is kicked in.

It is possible that the cropped image there contains more than one contour, for example, when another hand is close to the touching point and parts of that hand is cropped as well. In that case, I iterate all found contours. I render each contour in a black color to a temporary image with the same size to the cropped image. Next, I scan the touch blob area in the temporary image and sum up all

black pixels. In the end, the contour with the most black pixels inside the touch blob area will be considered as the touching hand contour.

```
// Input
//    newBlobs: new detected touch blobs
//    i: the processing blob's index
// Output
//    selectedContourIndex: the hand contour's index in the contourFinder object
function ExtractHandContour()
{
    int xpos = newBlobs->blobs[i].centroid.x;
    int ypos = newBlobs->blobs[i].centroid.y;
    int x = 0; int y = 0;

    // secure a valid size for the cropping image
    CalculateROI(xpos, ypos, getCamWidth(), getCamHeight(), x, y);
    thresholdImg = CropImageWithThresholdRange(rawImage, x, y, low_threshold,
         high_threshold);

    int nContour = contourFinder.findContours(thresholdImg, minHandSize, maxHandSize);
    int selectedContourIndex = 0;
    if (nContour > 1)
    {
        // to determine which contour is correct, we simply check which
        // contour that the touching blob resides in has the most pixels
        int maxPixels = 0;

        // blobs are sorted by size, the last one has the biggest size
        for (int i = contourFinder.blobs.size() - 1; i >= 0; i--)
        {
            // reset a dummy image to black
            dummyImg.set(255);

            // gets the total points of this contour
            CvPoint *points = new CvPoint[contourFinder.blobs[i].pts.size()];

            // render the contour points to black
            FillConvexPoly(dummyImg.getCvImage(), points);

            // cleanup resources
            delete []points;

            // now scan the touchingBlob's points, see how many pixels it
            // resides in this contour
            int pixelCounter= 0;

            for (int blobIndex = 0; blobIndex<touchingBlob.pts.size(); blobIndex++)
            {
                int base = CalculatePixelAddress( dummyImg );
                int touchPixel = pixels[base];

                if (touchPixel == 255)
                    pixelCounter ++;
            }

            if (pixelCounter > maxPixels)
            {
                selectedContourIndex = i;
            }
        }
    }
    return selectedContourIndex;
}
```

```
// Input
//    original: the raw image
//    x: the cropping top location
//    y: the cropping left location
//    lowThreashold: the lowest pixel value
//    highThreashold: the highest pixel value
// Output
//    dest: the cropped image
ofxCvGrayscaleImage CropImageWithThresholdRange(ofxCvGrayscaleImage original, int x, int
      y, int lowThreashold, int highThreshold)
{
    int width = 120;
    int height = 120;
    unsigned char * pixels = original.getPixels();
    int totalWidth = original.getWidth();
    int subRegionLength = width * height;
    unsigned char *subRegion = new unsigned char[subRegionLength];

    for (int i = y; i < y+height; i++)
    {
        for(int j = x; j < x+width; j++)
        {
            int base = (i * totalWidth) + j;
            int originalPixel =  pixels[base];
            if (originalPixel<lowThreashold || originalPixel > highThreshold)
            {
                originalPixel = 0;
            }
            else
            {
                originalPixel = 255; // set it to black
            }

            subRegion[result_pix] = originalPixel;
        }
    }

    dest.setFromPixels(subRegion, width, height);
    delete[] subRegion;

    return dest;
}
```

3.4.3 *Derive Finger Orientation*

In the first step to find the finger orientation, I draw two circles around the touching point. The first radius of the ellipse is selected using the larger value from the width and the height of the touching blob's bounding box; the second radius is the the first radius plus a value depending on how many times this method has been called. Next I remove all other pixels outside of the two circles. The remaining should be a contour, which is considered as the centroid of the

finger (shown in Figure 12). If such a contour cannot be found, I increase the radius and try again. The number of attempts is limited to 3 due to the performance concern. After that, I will just use the centroid of the extracted hand contour (from the previous ExtractHandContour method). In this case, the green line and the red line are overlap (Figure 12), which results in handedness undetermined. This situation could happen when the user's finger is perpendicular to the touch surface.

```
// Input
//    selectedContourIndex: the hand contour's index
//    thresholdImg: the cropped image
//    attempt: the number of attempts of using this method
// Output
//    angle: the angle between the touching point to the finger contour centroid
//    angle1: the angle between the touching point to the hand contour centroid
function DeriveFingerOrientation(int &attempt)
{
    ofxCvBlob handContour = contourFinder.blobs[selectedContourIndex];

    dummyImg = thresholdImg;

    float radius = max(touchingBlob.boundingRect.width, touchingBlob.boundingRect.height
        );

    // after the following 3 operations, dummyImg has only the yellow ring
    DrawFirstCircle(dummyImg, radius);
    DrawSecondCircle(dummyImg, radius + 5 * attempt);
    RemovePixelsOutOfTwoCircles(dummyImg);

    ofxCvContourFinder centerContourFinder;
    int nContour = centerContourFinder.findContours(dummyImg);
    if (nContour > 0)
    {
        if (attempt >= 3)
        {
            // this method uses the centroid of the extracted hand contour
            angle = angle1 = AngleBetween(handContour.centroid, touchingBlob.centroid);
        }
        else
        {
            fingerContour = contourFinder.blobs[0];

            // two angles are detected, so handedness can be properly checked later
            angle = AngleBetween(fingerContour.centroid, touchingBlob.centroid);
            angle1 = AngleBetween(handContour.centroid, touchingBlob.centroid);
        }
    }
}
```

4

EMPIRICAL VALIDATION

In this chapter, I describe in details the evaluation of finger orientation for touch discrimination. Three studies were conducted, including an exploratory study, a multi-user study with a simple tapping task, and a third multi-user study with a set of more complex tasks.

4.1 EXPLORATORY STUDY: IS FO DISTINCT ENOUGH?

This exploratory study investigates the distribution of 'natural' index finger placements across a tabletop and allow me to contrast the profiles of various standing positions around the table. Natural finger placement is referred to users' own finger touch postures without having any knowledge of how the system works. My goal is to discover if natural FO patterns are distinctive enough to be useful as a feature for user touch discrimination. I use the collected data as training samples for an SVM classifier to determine the potential accuracy rate for predicting user positions.

4.1.1 *Data Collection*

I collected finger orientation data for various user positions from one participant at a time. The tabletop was divided into an 8×8 grid, with each cell measuring 9.1×6.2 cm. The only instruction to the participants was to select targets, when they appeared, with users' right hand index finger. The targets were rectangular, measured 3.4×2.9 cm, and were placed at the center of a randomly selected grid cell. In the background I ran the FO algorithm and stored each orientation. When a target was hit, the color of the target was changed from red to green. I did not provide any additional visual feedback to indicate if the participant's finger orientation was correct or not.

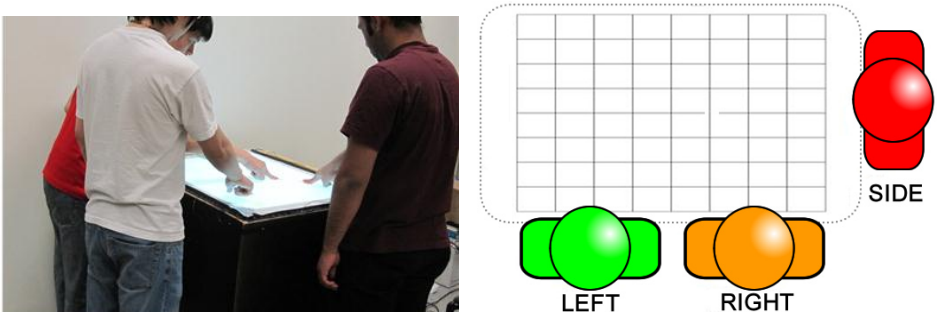


Figure 15: (a): Three Participants in an experiment around my custom-built FTIR tabletop; (b): dimensions of my system and the three positions for which I collected data to train my prototype system.

Participants selected a target in each cell, over two repetitions of all cells, while standing in each of three positions around the tabletop, LEFT, RIGHT, or SIDE (Figure 15b). I only collected data from these three positions, as all other major positions around the tabletop could be extrapolated from these (discussed in section 4.1.2). I collected

data from 8 participants × 3 positions × 64 target locations × 2 repetitions = 3072 trials. Each set of trials took approximately 45 minutes to complete.

Figure 16 shows the range of FO values for each cell in the grid. Each triangle represents the full range of finger orientations collected for the corresponding cell. The long midline depicts the mean value and the short line perpendicular to the midline shows one standard deviation from the mean. Following are some notable observations:

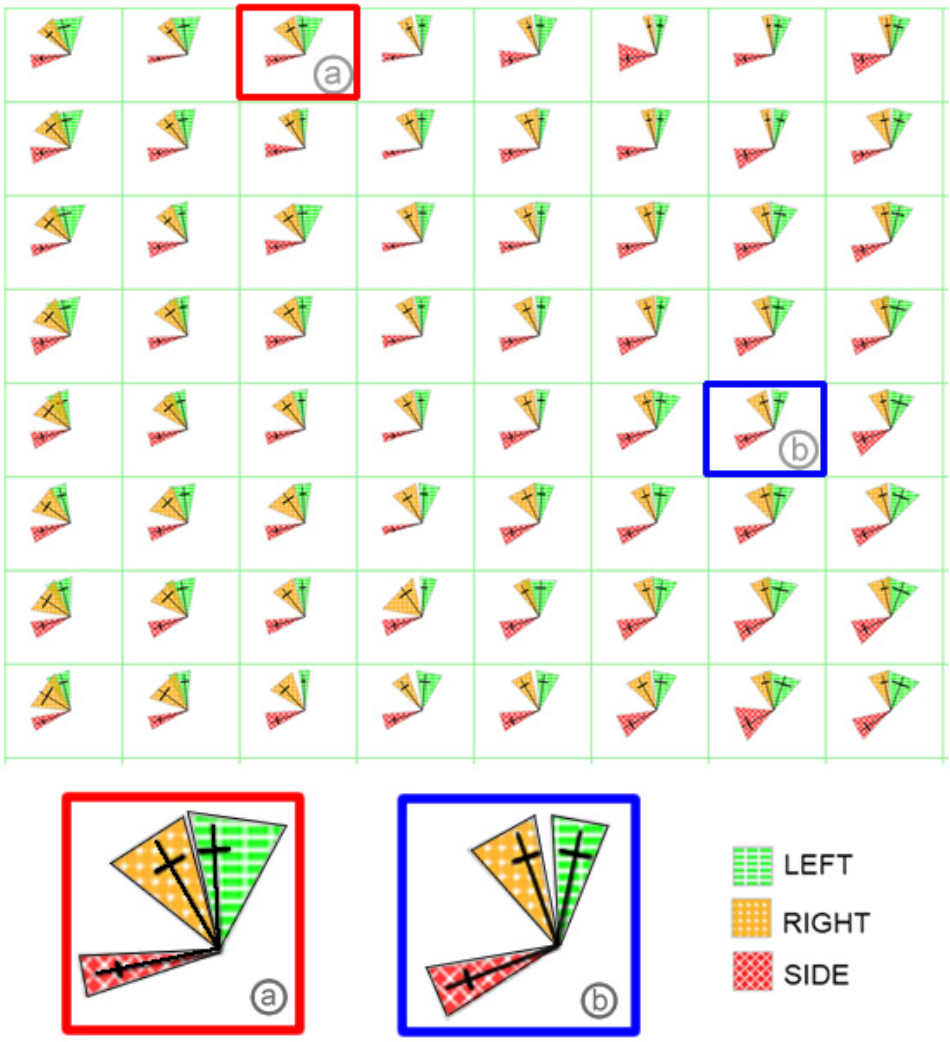


Figure 16: FO ranges across tabletop, with mean and standard deviation (Green: LEFT; Yellow: RIGHT; Red: SIDE). Two example cells are enlarged to show the distinct ranges.

- *Finger Orientation Ranges:* Surprisingly, over 80% of all cells exhibit very narrow standard deviations. In about 90% (58/64 cells) of cases, the mean angles fall approximately in the middle of the detected angle range. Cells in front of the user tend to have narrower ranges than those that are off to either side.

- *Range overlap:* The ranges exhibit very little overlap. The LEFT and RIGHT (green and yellow, respectively) positions are nearly shoulder-to-shoulder, likely a worst case scenario. Despite this very close proximity, finger orientation ranges are distinct in over 95% of cells for side-by-side positions and in all cells for orthogonal positions (i.e. SIDE vs. LEFT or RIGHT). The standard deviations of the ranges do not overlap in any situation.

- *Zones:* Overlap between ranges appears to be greater in regions of the table that are either further away from pairs of users. Thus for objects directly in front of a user, their finger orientation is more distinct than in shared territories further away. I consider this factor in my evaluation.

These findings stem from participants using only their right hand. A mixture of both left and right hands would inevitably show more variability. Since my FO algorithm can also detect handedness, I can first identify the handedness of a touch and then use the correct (left or right) profile to determine position. Therefore, I recruited the same group of participants and asked them to redo the tasks with their left hand index finger. Figure 17 depicts the results of the FO ranges for left hand.

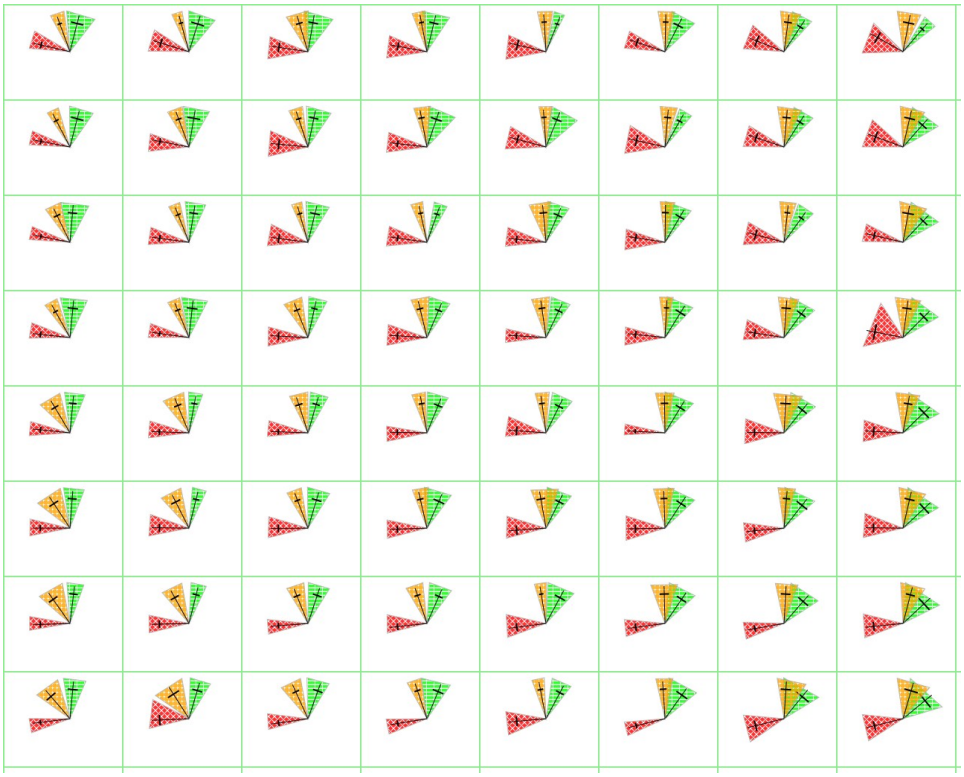


Figure 17: FO ranges from left hand index finger across tabletop, with mean and standard deviation (Green: LEFT; Yellow: RIGHT; Red: SIDE).

4.1.2 *Training Machine*

I classify FO patterns by user position using a multi-class support vector machine (SVM). SVM is a machine learning classifier that uses a set of training samples to create a mathematical function, or model, that can predict the correct category, or label, of an previously uncategorized item. I choose SVM because of its widespread reported success in a variety of problems. I used Chang and Lin's libSVM [2].

Training

To train the SVM, I collected user input data to create a set of labeled feature vectors (arrays of input values). My feature vector contains

the x-y coordinates of a touch and the corresponding FO angle, θ. For simplicity, I discretized the input space of the tabletop into 64 cells. The label of a feature vector is an integer representing the user's position around the table. Before training, I find the combination of required SVM parameters that give the highest cross-validation score. Since data were collected from both left and right hand index finger, I constructed two sets of training machines. One set is for the right hand index finger touch, another set is for the left hand index finger touch.

My model is user-independent, meaning that the training set includes data from multiple users and generalizes sufficiently to allow recognition of new users. User-independent systems are generally considered to be more difficult to implement than user-dependant systems, which are trained specifically to recognize one individual user. Even though I only collected data from three positions (LEFT, RIGHT, and SIDE, see Figure 15b), I can easily convert those three positions to other positions (see section 4.1.2).

Predicting a Touch's Owner

I used the SVM model to discriminate user touches when the tabletop camera sees a touch point. To trigger a prediction, I construct an unlabeled feature vector for a detected touch, consisting of the x-y coordinates of the finger and its orientation, θ. When the feature vector is fed into the SVM, it returns the value of the predicted user position. The first prediction is triggered when the touch happens. By doing so, the system is immediately aware of the owner of the touch. Once a touch is associated with an owner, no SVM prediction will

be triggered again. However, the touch's FO values will be updated every 50 frames (The camera runs 60 frames per second, so each 50-frame update is roughly 800 milliseconds). This approach can avoid running too many image process operations to overkill the system while still accurately keeping track of the touch's FO changes.

A predictive model is referred to a classifier that supports N classes, where a class is a user standing position. To simplify the problem, I start with a 3-class model, which classifies the three positions shown in Figure 15b. However, I can use data collected from a few positions to extrapolate to others, and combine them into various configurations (see section 4.2, below). Given the assumption that user pointing profiles are invariant to position, it is possible to take an alternative approach that generalizes to any possible user position, for example a user standing at a corner. Likewise, the inclusion of multiple fingers from a single hand is likely possible. Extensions to dynamic hand configurations and those involving more than 3 users are left for future work.

Profile Conversion

The tabletop I use can support up to six users working together. The full configuration is shown in Figure 18. As mentioned earlier, I collected data from only user 1, 2, and 3. For the remaining three positions, I follow this conversion logic: $3 \rightarrow 4$, $1 \rightarrow 6$, and $2 \rightarrow 5$. For example, when converting to position 4, I first create a set of new feature vector using the data of each cell from position 3, then change the finger orientation and match the cell from position 4's perspective, and finally update the label. It is important to correctly map one cell

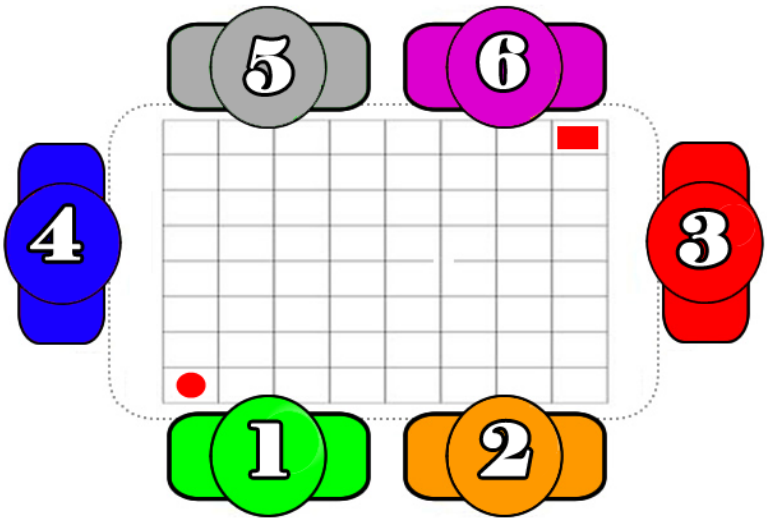


Figure 18: The table can support up to six users. User 3's cell (in red rectangle) is mapped as User 4's cell (in red ellipse)

to a different position's perspective. For instance, in Figure 18, from position 3's perspective, the cell in red rectangle should be mapped to the cell in red ellipse from position 4's perspective.

The following code snippet shows how to convert one position to another position. Each training data entity represents a feature vector, which includes target's location, finger orientation, and the user standing location (class label).

```
TrainingDataEntity ConvertEastToWest(TrainingDataEntity eastEntity)
{
    return new TrainingDataEntity()
    {
        TargetLocation = new Point(TouchTable.TableWidth - eastEntity.TargetLocation.X,
            TouchTable.TableHeight - eastEntity.TargetLocation.Y),
        FingerAngle = MathHelper.EnsureAngleIn360(eastEntity.FingerAngle - 180),
        StandingLocation = Framework.ReferenceData.UserStandingLocation.WEST
    };
}

TrainingDataEntity ConvertSouthRightToNorthLeft(TrainingDataEntity southRight)
{
    return new TrainingDataEntity()
    {
        TargetLocation = new Point(TouchTable.TableWidth - southRight.TargetLocation.X,
            TouchTable.TableHeight - southRight.TargetLocation.Y),
        FingerAngle = MathHelper.EnsureAngleIn360(southRight.FingerAngle + 180),
        StandingLocation = Framework.ReferenceData.UserStandingLocation.NORTH_LEFT
    };
}

TrainingDataEntity ConvertSouthLeftToNorthRight(TrainingDataEntity southLeft)
{
    return new TrainingDataEntity()
    {
        TargetLocation = new Point(TouchTable.TableWidth - southLeft.TargetLocation.X,
            TouchTable.TableHeight - southLeft.TargetLocation.Y),
        FingerAngle = MathHelper.EnsureAngleIn360(southLeft.FingerAngle + 180),
        StandingLocation = Framework.ReferenceData.UserStandingLocation.NORTH_RIGHT
    };
}
```

Cross Validation

The purpose of cross validation is to determine if the model can really work as I expect without actually running a user study. Cross validation techniques usually involve partitioning the sample data, constructing the SVM using one subset (training set) and validating with another subset (testing set). I use the following two approaches to cross validate my collected data (from eight users). In the first approach, I iterate eight users and build an SVM with seven users' data as the training set, then use the remaining user's data as the testing set. In the second approach, I construct the SVM using all eight users' data, randomly exclude a few records from each user, and then use excluded data form a testing set.

User ID	Accuracy			
1	96.88%	93.23%	94.01%	93.75%
2	95.05%	92.45%	92.71%	92.71%
3	97.66%	95.83%	96.09%	96.09%
4	89.06%	90.36%	91.15%	91.93%
5	93.23%	93.75%	93.49%	93.49%
6		86.98%	86.98%	88.02%
7			99.48%	99.48%
8				100.00%
Average	94.4%	92.1%	93.4%	94.4%

Table 1: Accuracy of 5-User SVM (column 2), 6-User SVM (column 3), 7-User SVM (column 4), and 8-User SVM (column 5).

First, I use the first approach to cross validate. Table 1 shows the accuracy of using 5, 6, 7, and 8 participants' data in SVM. Each row in the table represents using the user N as the testing set to evaluate the SVM built by the remaining users. Interestingly, when using only 5 participants to build the SVM, user 4 seems to be an outlier. Adding one more participant increases the user 4's accuracy but lower the overall accuracy and now user 6 becomes the new outlier. Adding another participant (seven in total) increases the overall accuracy and again improves the user 4's accuracy; however, it does not affect the user 6 at all. Finally, with eight participants, both overall accuracy (94.4%) and user 6's accuracy are improved. Moreover, half users' accuracy (2, 3, 5, and 7) remain the same compared to the 7-User. To better determine if the user 6 is really an outlier, I build a 7-User SVM without user 6 (see table 2), which lead an increase in overall accuracy, but only three individuals of the seven users have the accuracy increased.

I also use the second approach for data cross validation. Table 3 shows the results. Each row represents the accuracy of randomly selecting 2 to 10 records from each participants to form the testing set to evaluate the SVM built by the remaining data. The accuracy will depend on what records are randomly selected. Nevertheless, after running the program numerous times, the accuracy is consistently around 96%. Combining the results from both approaches, I believe that using eight participants to build the SVM is reasonable and the prediction accuracy is high enough for my purpose.

Two observations can be made from the results of cross validation. First, different users have different gesture of pointing and this provides a variety of samples in the SVM. Second, in my case, SVM provides a better result using the second data cross validation approach. These two observations make me believe that there is no reason to exclude user 6 and the SVM will perform better when using all users' data.

User ID	Accuracy
1	96.88%
2	95.05%
3	96.61%
4	90.10%
5	93.49%
7	99.22%
8	100.00%
Average	95.9%

Table 2: Accuracy of 7-User SVM Without the User 6

Random Trials Num	Accuracy
2	100.00%
3	100.00%
4	96.74%
5	96.52%
6	97.10%
7	95.65%
8	95.11%
9	96.62%
10	94.78%
Average	96.94%

Table 3: Accuracy of Cross Validation Using Approach Two;
First column means when building the SVM, I randomly exclude n records from each participant and use those data as the testing set.

The results of the two cross validation methods show a high accuracy of the prediction (94.4% and ≈96%). However, in a software-based cross validation, the testing set contains only correct data. In a real multi-user application, the system might receive a false inaccurate data. For example, one user may try to avoid physical contact from another user when they both operate at the same time. The user's finger data might not be as 'natural' as when operating alone. Therefore, testing with a multi-user experiments will be the next step.

4.1.3 *Hypotheses*

A multi-user scenario refers to multiple users working around a tabletop. To better design what I want to measure from the system, I first postulate the following hypotheses based on the observations from the collected data:

H1: Because of differences in range overlaps, SMESY will report higher accuracies for configurations where users stand in opposite or orthogonal positions than when standing adjacent to one another;

H2: Although training data were collected for targets at the center of each cell, the classifier will generalize across the entire cell, keeping accuracy high for unrestricted target positions;

H3: Since data were collected for a selection task, other tasks (such as rotating or scaling) that require users to place their fingers along different orientations will not be as accurate;

H4: Increased overlapping in cells that are more distant from a pair of users will lead to lower accuracy rates in those regions.

I evaluated these four hypotheses in the following two experiments.

4.2 STUDY 2: ACCURACY IN A REAL SETTING

This study examined the accuracy of SMESY with a tapping task, common on tabletops for triggering a command or object selection. I wanted to test the robustness of my system with multiple users in a variety of possible configurations.

4.2.1 *Participants and Procedure*

Eight groups of 3 participants each, between the ages of 20 to 35, participated in the study. Five of the 24 participants were female and all were right-handed. None had prior experience using a tabletop or participated in the exploratory study.

The task was identical to the pointing task used in the exploratory study, with two exceptions. First, target positions were not restricted to the center of a grid cell; and second, the task was performed in groups, arranged in 1 of 4 predetermined standing configurations.

Each participant was assigned to a unique color, and their desired target was displayed in that color. Similar to before, I asked the participants to select the target with their right hand index finger. Additionally, I encouraged participants to interact simultaneously. However, I did not explain to the participants what I were testing and how the system worked. Participants were told that I wanted to observe how multiple users naturally interact with a touch table.

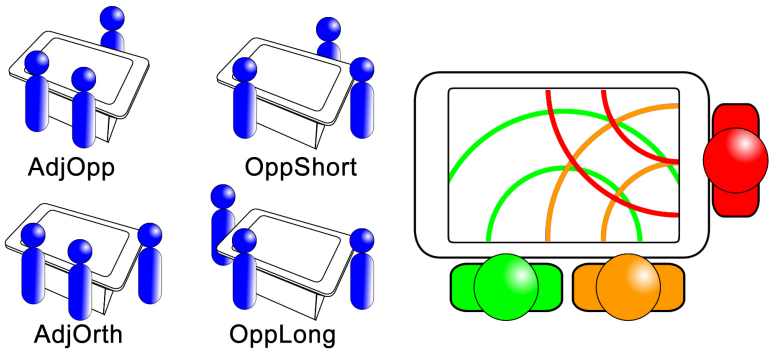


Figure 19: Left: Four configurations of standing positions relative to the table including side-by-side, opposite and adjacent users. Right: Targets were placed in 3 zones outlined by the same color as a user. The zones demarcated areas based on the distance to the right hand of each user.

4.2.2 *Design*

The experiment employed a 4 × 3 factorial design. The independent variables were Configuration and Zone:

Configuration: I chose a diversity of configurations that might appear in realistic situations. These include adjacent (side-by-side), opposite (across the long and short dimensions of the table), and orthogonal placements. The 4 configurations are labeled AdjOpp, AdjOrth, OppLong and OppShort (Figure 19 left).

Zone: The findings from the exploratory study showed an increase in range overlap as the distance from a pair of users grows. Therefore, I also tested my algorithm's accuracy based on the location of targets relative to each user's position. I defined 3 zones based on the distance to the user's right shoulder. The 3 zones are near (0–25cm), middle (26–45cm) and far (45cm to the end of table) (Figure 19 right).

I presented an equal number of trials in each zone and for each participant. The Configurations were counter-balanced to reduce any learning effect. For each trial, a target was placed in a randomly chosen zone. There were a total of 12 targets per user in each configuration. The design can be summarized as 4 Configurations × 3 Zones × 12 Trials × 8 Groups of 3 users = 3456 trials in total.

4.2.3 *Results and Discussion*

System Accuracy

The recorded data were analyzed using a repeated measures ANOVA test. The results, summarized in figure 20, revealed an average accuracy of 97.86% across all the tested conditions. I found no significant effect of Configuration ($F_{3,21}$ = 0.858, p = 0.48) or Zone ($F_{2,14}$ = 3.47, p = 0.65), thus rejecting **H1** and **H4**. In **H4**, I hypothesized that SMESY's prediction accuracy would decrease in far-away regions, which showed more overlap between finger orientations. The results show that this is not the case. Likewise, **H1** can be rejected, since results were not significantly affected by user placement.

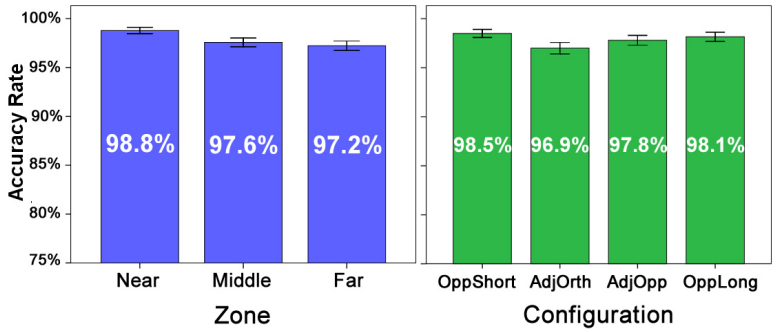


Figure 20: System accuracy based on zones (left) and configurations (right). Error bars represent 1 s.e. Scale starts at 75%, to show differences.

By Configuration, the accuracy rates were 98.5% (s.e. 0.4%) for OppShort, 96.9% (s.e. 0.9%) for AdjOrth, 97.8% (s.e. 1%) for AdjOpp and 98.1% (s.e. 0.9%) for OppLong. Notice that the accuracy of AdjOrth was slightly, but not significantly, lower than the others (Figure 19, 2nd from left). This was because there were more overlaps in finger orientations when participants were standing in this configuration.

It shall be noted again that no participant is aware of that the underlying goal of the tasks is to collect their finger orientation. All participants performed the tasks (a simple tapping) in their own natural way without being instructed. Hence, I consider the average accuracy of 97.86% a very good result.

Handedness Detection Accuracy

Because in this study, all participants were asked to perform the tasks with their right hand index finger, I can use the data to determine how well the handedness detection algorithm works. The results show 3154 out of 3456 trials (91.26%) were accurately detected as right hand. In those 3154 trials, 98.6% were correctly predicted as the correct user. In the 302 trials that were detected as left hand, 89.7% were correctly predicted. Interestingly, I later fed the remaining 10.3% (31) trials into a left-hand SVM, they were all predicted correctly as the desired users. This result inspires me to design the next study that allows users to touch with either hand, and the application dynamically switches to a different training machine according to the detected handedness (see section 4.3).

Observation in Groups

A significant effect was found in groups ($F_{7,3455} = 5.372$, $p < 0.01$), which implies that different groups behaved differently. However, no pattern was found among groups in either zone or configuration. Figure 21 and 22 show the results of accuracy of each group in zones and in configurations. Only three groups underperformed in the far zone and this indicated that distance of target does not impact the

accuracy. Five groups underperformed in the AdjOrth configuration, which confirmed the result in Figure 20 that adjacent users would impact the accuracy.

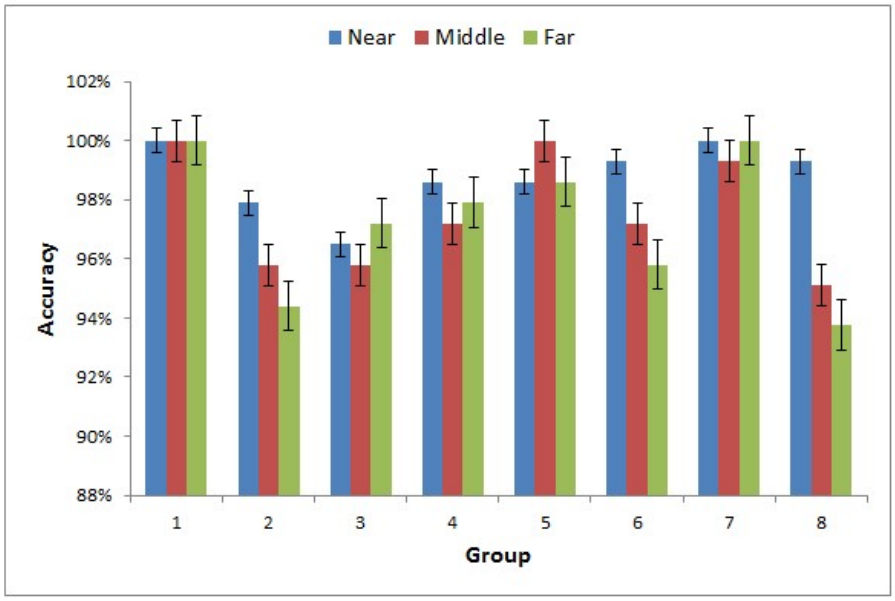


Figure 21: Accuracy of each group in zones.

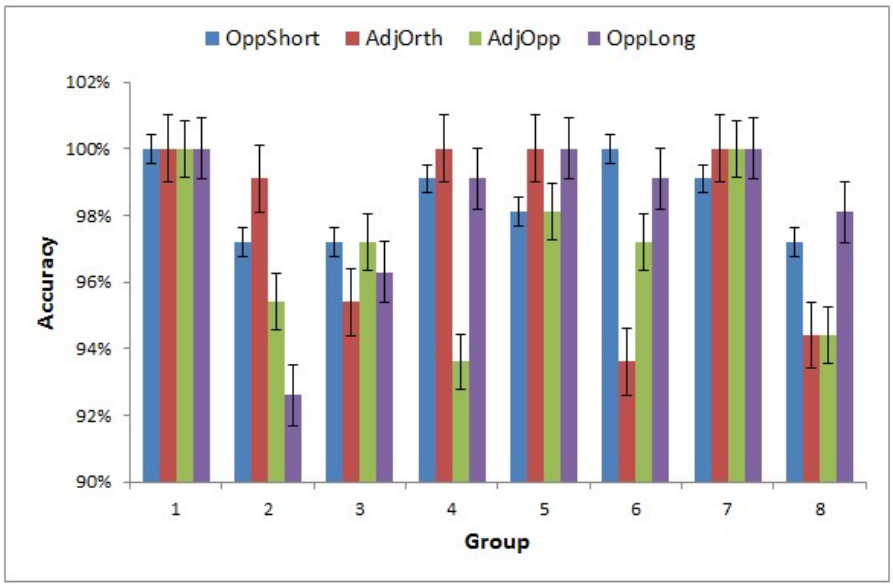


Figure 22: Accuracy of each group in configurations.

Inspection of Errors

In inspecting the errors from the current study, I observed that many were caused by a failure of my finger orientation algorithm. Because my system uses overhead lighting to produce hand contours, some group situations can result in problematic overlapping shadows, for example, when a user's finger is occluded by a neighbour's arm. Figure 23 shows two such situations in which hand contour extraction failed.

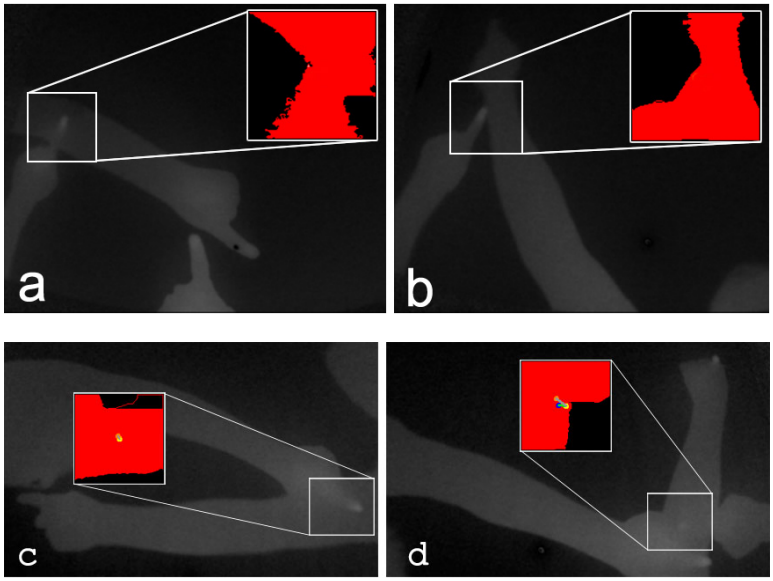


Figure 23: When a tap occurs (a) inside or (b) nearby the shadow of the other user's arm or hand, the algorithm failed to detect the correct FO. (c) and (d) demonstrate other two similar scenarios.

To further explore why the errors occurred, I manually went through the 74 trials, out of 3456 trials that fail to be predicted as the correct user. Because every target belongs to a certain cell (defined in Figure 16) and I have collected the desired FO mean and its standard deviation. I identify a user's FO as an outlier if the FO value differs by twice the standard deviation or more from the

mean. I found that 63 out of 74 trials are identified as outliers. When looking at the recorded raw images one by one for those 63 trials, the incorrect FOs are all caused by overlapped arms (depicted in Figure 23). The remaining 11 trials are shown in table 4. These trials were predicted as a incorrect user even the FO fell in a reasonable desired range. Interestingly, 10 of them come from the configuration that include adjacent users (AdjOrth and AdjOpp). The data shows that configurations with adjacent users might have a higher chance to cause the SVM to predict inaccurately, but 10 out of 1728 trials (2 Configurations $\times$ 3 Zones $\times$ 12 Trials $\times$ 8 Groups of 3 users), which is less than 0.01, is not significant. Hence, I believe that it is promising to use FO for touch discrimination with multiple users standing in opposite or orthogonal positions. In addition, having only two adjacent users on one side of the table will still work reasonably well. However, I assume that if there are more than two adjacent users on the same side, this technique might be less accurate.

Config	Desired User	Predicted User	User FO	Range Mean	Range S.D.
AdjOrth	1	2	125.93	103.18	14.07
AdjOrth	1	2	114.46	97.65	11.3
AdjOrth	1	2	118.04	104.4	14.68
AdjOrth	1	2	118.34	100.83	14.06
AdjOrth	2	1	105.43	125.08	10.96
AdjOrth	1	2	98.86	86.52	6.73
OppLong	4	2	90	26.41	7.34
AdjOpp	1	2	105.23	84.78	10.83
AdjOpp	1	2	109.22	97.65	11.33
AdjOpp	1	2	114.6	97.65	11.33
AdjOpp	1	2	124.51	104.4	14.68

Table 4: The FO in these 11 trials are considered as a valid angle for the FO range at the desired target; however, they are predicted as a different user (user labels are defined in Figure 18). This kind of error is considered as prediction error from SVM.

4.3 STUDY 3: STEP UP COMPLEXITY

The previous study showed that SMESY is highly accurate across multiple user positions and when the targets are placed across the display, but only demonstrated this for the case of selecting objects. Real-world applications often involve more complex tasks. For instance, a user may want to rotate and scale a picture or draw on the table. These tasks may involve using both hands or may lead users to touch the table in a different orientation. SMESY relies solely on users' touch orientation. Prediction errors can result with users' changing their touching behaviour, whether intentional or subconscious. However, I hypothesized that this issue could be resolved by educating users about how the system works so that they can

adapt themselves to the system. I further hypothesized that such adaptation is effortless and welcomed by the users.

4.3.1 *Participants and Procedure*

I recruited 9 groups of 3 participants, each between the ages of 20 and 35, for this study. All 27 participants were right handed and 2 were female. Five had participated in Study 2, but none participated in the initial data collection study.

I tested my system using three tasks involving the manipulation of a 7.8 $\times$ 9.8 cm (240 $\times$ 180 pixel) object:

1. *Rotation with right hand (RR):* Rotating an object is likely to produce some finger orientations (on land down) that do not coincide with what I used for training my algorithm. In this task, participants were restricted to using their right hand only.

2. *Rotation with either hand (RE):* This is the same task as the one above except that participants were allowed to use either hand to rotate the object. This could test my algorithm accuracy of handedness detection.

3. *Scaling (S):* This task requires participants to use both of their index fingers to tap on a rectangular object, and drag in opposite directions. This task would further test the limits of my trained system as well as the accuracy of my handedness detection algorithm.

In task RR, hand prediction is unnecessary and thus all inputs were passed to only the right hand FO model, allowing me to evaluate my

handedness detection algorithm. The RE and S tasks do not restrict users' hand and thus test the system under more realistic conditions. For these tasks, I loaded a second model using left hand data. All inputs were first evaluated for handedness and then passed to the appropriate model for user touch discrimination.

4.3.2 *Design*

The experiment consisted of 2 phases. The 1^{st} phase imitated a walk-up-and-use scenario, where participants performed the 3 tasks without any knowledge of how the system works. The 2^{nd} phase started with a short orientation session (about 5 minutes long), where participants were informed about how the system worked. In the 2^{nd} phase only, participants received feedback during the 3 tasks about whether the system correctly recognized them. A colored arrow was shown, along with a smiley face for correct predictions, or a sad face for incorrect ones. Participants were given practice trials until they understood the meaning of the feedback and had learned to avoid situations that commonly caused recognition failure, such as shadow occlusion or extreme FO angles.

Participants were asked to stand in the AdjOrth configuration, which produced the lowest accuracy in Study 2. In each trial, 3 targets, color-coded by user, were placed in random positions. A small offset distance was used to ensure that targets did not overlap with each other or appear too close to the edge of the table.

The experiment employed a 3×2 within-subject factorial design. The independent variables were Task (RR, RE, and S); and Feedback

(feedback or non-feedback). Task was partially counter balanced, however the non-feedback phase was always presented first. I allowed short breaks between tasks and phases. Participants filled out a questionnaire upon completion.

4.3.3 *Results and Discussions*

For all the 3 tasks, the recognition of user position was made based on the initial touch of an object. For the scaling task, I used the FO from whichever hand touched the object first. The resulting data were analyzed using Repeated-Measures ANOVA and Bonferroni corrections for pair-wise comparisons. The results revealed an average accuracy of 94.7% across all the tested conditions. ANOVA tests yielded a significant effect of Feedback ($F_{1,8} = 5.7$, $p < 0.05$). There was no significant effect of Task ($F_{2,16} = 0.74$, $p = 0.49$).

The system had higher accuracy in the feedback condition (96.5%, s.e. 0.4%) than in the non-feedback condition (92.8%, s.e. 1.5%). I found no significant learning effect during the 1^{st} phase, suggesting that this difference was primarily due to the following orientation session. When broken down by task, I find accuracies of 95.8% (s.e. 0.6%), 94.4% (s.e. 1.8%), and 93.7% (s.e. 1.1%) for RR, RE and S, respectively.

Effect of Task Complexity

Although analysis did not yield a significant effect of task complexity, one-way ANOVA tests showed a significant difference between the 3 tasks in the non-feedback condition ($F_{2,1941} = 4.57$, $p < 0.05$). RR

had the highest accuracy (95.1%, s.e. 0.9%), followed by RE (92.6%, s.e. 1%), which was higher than S (90.7%, s.e. 1.1%) (Figure 24 left). Post-hoc analyses showed only a significant difference between RR and S ($p < 0.01$).

Many of the errors in the 1st phase were a result of overlapping shadows that interfered with FO detection (as shown in Figure 24). Task S had the highest number of these errors because 2 hands per user resulted in more overlapping arms. Additionally, in this task, users would often place their hands with the index finger parallel to an object's edge to avoid occlusion. In addition, RR tasks in the non-feedback condition is actually similar to the tapping tasks in study 2. Participants had no knowledge of the system and performed the tasks in their own natural way, but the accuracy was reduced to 95.1% (RR in study 3) from 96.9% (AdjOrth accuracy in study 2). As predicted, the system accuracy decreased with increasing task complexity (between RR and S), confirming **H3**. I assume that the knowledge and feedback reduced this effect in phase 2.

Effect of Feedback

In the feedback condition, system accuracy increased to 96.6% (s.e. 0.7%), 96.3% (s.e. 0.7%), and 96.6% (s.e. 0.7%) for RR, RE, and S, respectively (Figure 24). Pairwise comparisons showed a significant improvement over the non-feedback condition for all the tasks except RR ($p < 0.01$). These results suggest that by understanding the causes and recognizing instances of problems, such as FO detection errors, users were able to adapt and improve their experience.

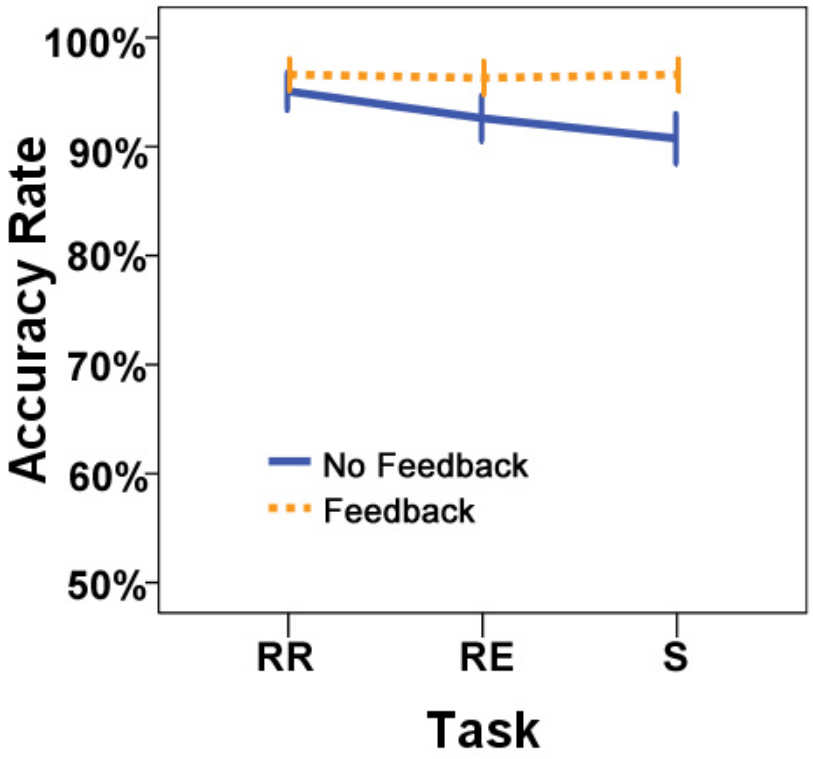


Figure 24: System accuracy shown by task and feedback (graph starts at 50%).

Handedness Detection Accuracy

For evaluation of handedness detection, I use only trials from the RR task, in which hand use was controlled. In this task, the right hand was correctly determined 93.8% of the time (Figure **??**). The number is close to the accuracy of handedness prediction (91.26%) in the study 2. Therefore, I believe it is reasonable to expect a similar accuracy for detecting the left hand. Within the set of trials for which handedness was correctly recognized, user positions were also predicted correctly in 95.6% of cases. Interestingly, even when handedness detection failed, user identification remained high at 91.3%.

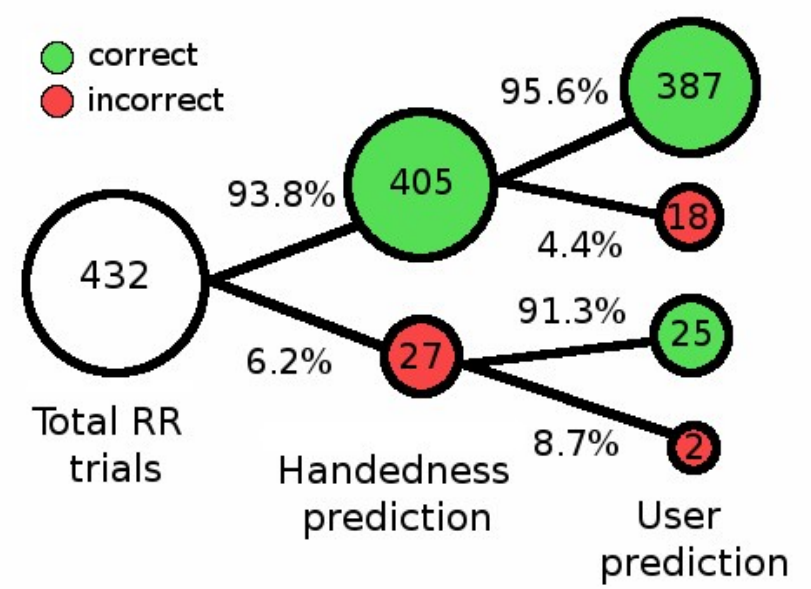


Figure 25: System accuracy for hand and user predictions. User prediction is still high even when handedness prediction fails. (RR task only)

Learning Among Groups

After breaking down feedback conditions with groups, I notice seven groups have significant improvement in the with-feedback condition (see Figure 26) and one group (group 7) performs equally in both conditions. Only one group (group 6) have the reverse effect; however, their accuracy remained high (over 95%). Another interesting observation is, before training, participants all have similar long standard deviation, which indicates without knowing how the system works, users can produce very different and dramatic results. But after users gain knowledge of the system, their standard deviation become much shorter, which means they produce much stable and consistent results.

Among these nine groups of participants, I observed a behaviour change after educating about them how the system worked. Some

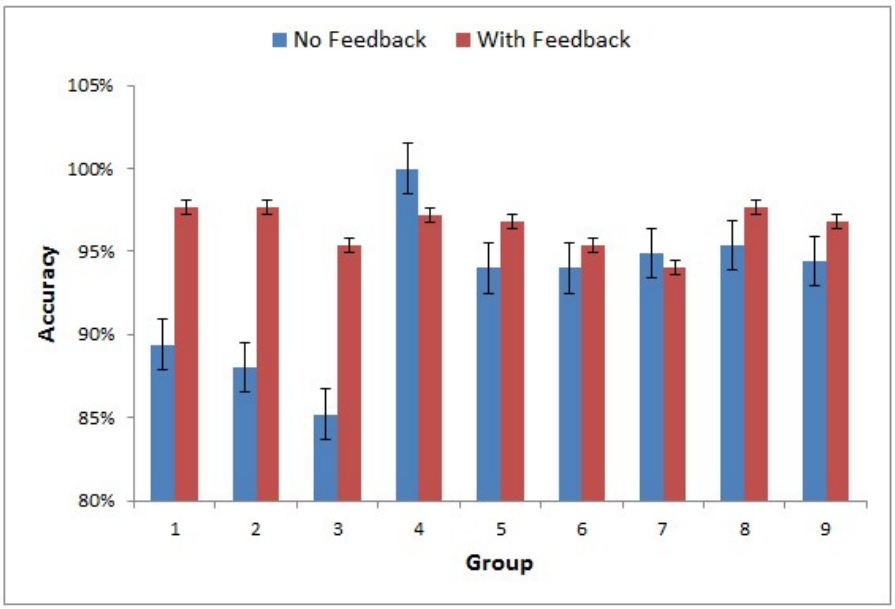


Figure 26: Accuracy were improved in seven groups after training and providing feedback in tasks.

participants tended to be more careful when interacting with the system after knowing what they could do to mess it up. I assume that some users might make more mistakes when they are fear of making mistakes. This potential psychological phenomenon is beyond my study. In spite of that, I believe the majority of users will achieve a higher accuracy after learning how the system works. Furthermore, for some users, their natural way of interaction can also yield a high accuracy even without knowing how the system works.

Subjective Preference

The post-experiment questionnaire shows that users welcome SMESY as an easy-to-use plug-in for existing tabletop applications. All scores reported below are based on a 5-point Likert scale (5 for highest preference).

The participants gave an average score of 4 in support of user feedback. Of all participants, 85% agreed that the feedback helped them learn from mistakes, and better adapt to the system. When asked “Did you change your finger direction after knowing how the system worked?", they responded with an average of 3.1. They reported an average of 1.8 when asked if they felt it was uncomfortable to change their FO. In most cases, however, such a change was not necessary; only 1 user (3.7%) gave a positive score (of 4) when asked if the required number of corrections was excessive. Participants also gave feedback regarding my UI design, with 78% in support of showing the detected FO in addition to the visual feedback of the recognition result. This motivated my design of the Position Aware Cursor, which I describe in the following chapter.

5

ENHANCEMENTS AND DISCUSSIONS

The results of my studies suggest that the robustness of SMESY will allow the support some multi-user features on a common tabletops. The results also inspire me to design two additional enhancements to SMESY. The first allows users to move around the table and the second allows for a fluid method of correcting prediction errors. Both features are compatible with the lightweight requirements outlined earlier.

5.1 USER MOBILITY

To grant users the flexibility of moving around the table, I associate each user with a Position Avatar, shown in figure 27. Users log in to the system by selecting a Position Avatar icon. Thereafter, the icon indicates their position at the tabletop edge. When a user chooses to changes positions, she can drag the Position Avatar along. In this implementation, the onus is on the user to manually inform the system of their movements. Although a more sophisticated device could automatically track the user with peripheral hardware (discussed in section 6.1), I resorted to manual placement to maintain the lightweight nature of SMESY.

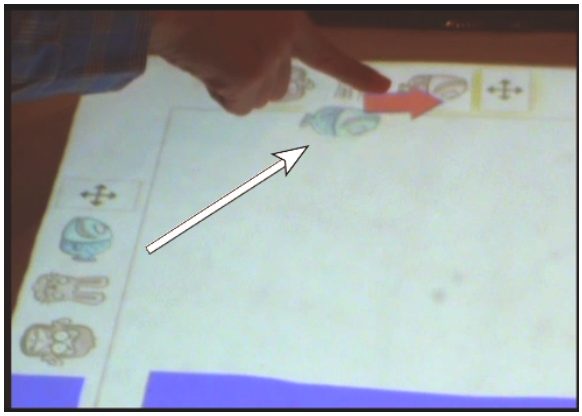


Figure 27: Position Avatar: A user drags her avatar (fish) to another position and her profile, suck as selected thickness and color, are transferred to the new location.

5.2 FLUID ERROR RECOVERY

Error recovery is an instrumental feature of a lightweight system, as it may not always guarantee 100% accuracy. Inspired by comments from my participants, I designed the Position Aware Cursor (PAC, Figure 28) to provide users with a fluid and robust solution in cases of wrong predictions. PAC has two elements: (1) A color-coded arrow showing the user's touch orientation, and (2) a set of wedges showing the possible FO ranges available, based on the locations of other users. In this example, the angle and direction of these wedges are based on the data collected in my exploratory study (Figure 16). If an incorrect prediction occurs, the user can reorient her finger to a new wedge. Figure 28 left shows that a user lands her finger and the system displays her location. Then the user reorient her finger (Figure 28 right) to change her position identity. I envision that such a feature could be disabled when a user becomes acquainted with the technique.

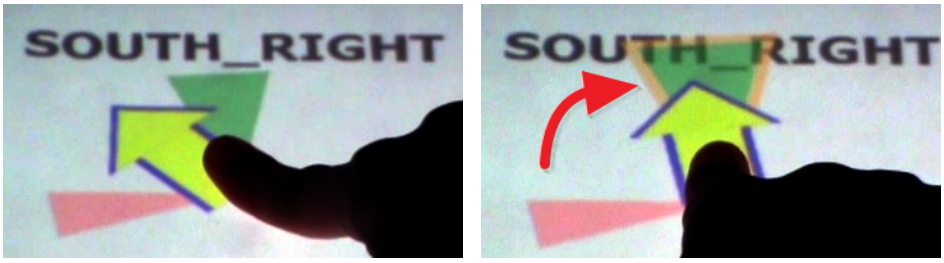


Figure 28: The Position Aware Cursor. Left: a user lands her finger, and the system predicts her location correctly. Right: the user reorient her finger to change her position identity.

5.3 USER SUBJECTIVE IMPRESSION

In a final informal evaluation I collected subjective user feedback with SMESY in two prototype applications: a multi-user paint application and a mahjong solitaire game.

Three groups of 3 participants (2 females), between the ages of 21 and 30, participated in this evaluation. Before each group started the tasks, I explained how the system works and the use of PAC. I also let each participant try PAC for a few times and made sure each of them understand it. However, I did not present the Position Avatar to them. With the paint application, participants were asked to collaborate and replicate a sample drawing. This required that they each control certain user-specific states such as line thickness and color. Each participant completed $^1/_3$ of the drawing. In the multi-user game, participants were asked to quickly find and select two tiles with matching graphical patterns. Tiles could occlude and overlap one another, thus requiring participants to move tiles around the table. Users were given a score based on the number of pairs

they matched and the game ended when all tiles were selected. I encouraged participants to use PAC for error corrections.

I note the following observations: (1) Participants finished the tasks relatively quickly, and were not hindered by any system features. (2) In informal interviews, participants indicated that they appreciated the multi-user capabilities of SMESY, and mentioned that they preferred the simultaneous operations to taking turns to carry out the same tasks. (3) They appreciated that they were not required to wear peripherals or hold a pen for user identification. (4) Two participants mentioned that they used PAC to correct errors. (5) Participants found that PAC helped them understand the method by which the system associated touch with user position. (6) Interestingly, one participant commented that the only concern he had with SMESY was the inability to move from one position to another. I then allowed him to try out the Position Avatar, of which he reported satisfaction. (7) Two participants from one group suggested that such a system could be implemented by recognizing their fingerprints. Given the technical challenges and hardware requirements for fingerprint recognition with current technology, SMESY is an ideal alternative for distinguishing multiple users' touches.

5.4 DISCUSSIONS

Overall, my results are highly encouraging and confirm the potential of SMESY as a viable approach for multi-user capabilities on common vision-based tabletop systems.

5.4.1 *Findings*

SMESY is the first system that tries to integrate Finger Orientation with a multi-touch tabletop for touch discrimination purpose. I evaluate the system in two separate controlled multi-user studies and one small qualitative study. I highlight some of my primary findings from all studies:

Reliability across entire tabletop Overall, the SVM classifier is robust in my applications. Although my training set is collected on only 64 target locations, the system is able to classify interactions across the entire continuous table space (confirming **H2**).

Handedness Even though the detection of handedness is not extremely accurate, the user prediction accuracy remains high overall. The majority of samples for which the hand is identified incorrectly are still associated with the correct user position.

Accuracy across tasks SMESY responds well to untrained finger orientations that result from non-pointing tasks as well as from awkward approaches when users reach around one another during simultaneous interaction. Feedback further improves the prediction accuracy.

Generalizing to users The system easily generalizes to new users who did not contribute to the training data set. This type of generalization is typically a difficult problem in machine learning, but is possible in my approach because of the distinct ranges of FO values, even across multiple users.

User configurations As expected, there is a slight penalty in prediction accuracy for adjacent users sharing a table edge. This is due to adjacent users exhibiting the most amount of overlap in FO. However, the loss was smaller than I expected as I did not find any significant differences in accuracy across different user configurations.

User adaptation Another interesting observation was the willingness and ability for users to adapt to the system. I found higher success rates when users were told how the system operates. Users were comfortable in altering their finger landing orientation to make the system work even more effectively. Users also reported that they did not feel any additional cognitive or motor effort than when they were not given any system knowledge. Furthermore, groups displayed an eagerness to cooperate, by adjusting their hand position to make room for others and by taking turns when simultaneous selection was impractical, thus exhibiting common courtesy.

Complementarity SMESY could work as either a stand-alone system or one that could be used in conjunction with other methods, as in [4, 6, 8]. For example ceiling mounted cameras can provide some information about users interacting around a tabletop. In areas of high occlusion, where cameras may not properly detect certain actions, the system could resort to using SMESY.

5.4.2 *Implications*

My exploration of FO profiles highlights some important implications for designers: First, people appear to produce very consistent finger orientations for pointing tasks. Although different groups of people show diversity, their FO falls in a certain range within a restricted demographics. Therefore, FO is the easiest and natural way to distinguish in pointing tasks, but is also reliable in more complex situations. Moreover, there is also potential for FO in contexts other than user discrimination.

Second, users standing at locations that are on orthogonal and opposite sides of the table can be distinguished with a very high reliability. I believe that using FO for touch discrimination will be highly reliable in smaller touch devices, such as a coffee table, a multi-touch tablet, or even a multi-touch smartphone. For example, multiple players can sit in each side of the device to play competitive games. One user per side is an ideal configuration, but my studies show that having two adjacent users on the same side can yield a reasonably high accuracy as well. Therefore, designers can be confident that using FO for touch discrimination with one user per side and shall not deter from using this feature in more crowded conditions.

Next, although once users understand how the system differentiates their touches from others, their could try to fake the touch angle to act as a different person, impersonation is not considered as a limitation of my system. In a collaboration scenario, multiple users try to complete a task together. There is no need to impersonate

because their ultimate goals are the same. In contrast, in a competition scenario such as games, impersonation is less concerned for two reasons: 1) players have integrity. Since all players are presenting themselves, if one player is cheating, it is likely that he or she will not be welcome in the future games; 2) in some games players only want to gain points for themselves instead of giving points to opponents. Hence, players will want to be identified correctly by the system.

Finally, even though my FO algorithm is implemented with a modified FTIR table, it will work in the same way with any system that is able to capture the touch hand contour. When using FO to discriminate touches, the Position Aware Cursor is a fluid and easily implementable feature that can not only improve the reliability and robustness of a touch-discriminate system, but also quickly help users understand how the system works.

6

SUMMARY AND FUTURE WORK

6.1 CONCLUSION

Differentiating user touches is a feature that is currently missing in all multi-touch devices. In this thesis, I first report on investigations of using finger orientation to discriminate user touches. Next I characterize the qualities of lightweight touch discrimination techniques. And finally I describe my new technique, SMESY, that uses finger orientation only to identify the touching user's standing location, which I defined as Position Awareness. SMESY has specific advantages that makes it more appealing than other existing methods.

I describe in details the design and the implementation of SMESY as well as the results and analysis of three user studies intended to test the accuracy, reliability and satisfaction of this technique. To my knowledge, my research is the first one to empirically evaluate using finger orientation only for touch discrimination purpose with multiple users. My studies show finger orientation is a simple, flexible and accurate method to discriminate user touches on tabletops. Furthermore, I introduce two enhancement techniques for multi-user applications: Position Avatar and Position Aware Cursor. With these two enhancements, users can change locations and perform

self-correcting actions in a fluid manner, without interrupting their activities.

In conclusion, SMESY is a viable lightweight solution for providing simple yet effective support for multi-user application features on tabletops. The main contributions of this research are:

- The summary of existing techniques to differentiate user touches on tabletops.
- The description of a new simple finger orientation algorithm.
- The classification of lightweight touch discrimination techniques.
- The introduction of SMESY for position awareness (PA) around a multi-touch tabletop.
- A thorough evaluation and analysis of SMESY with multiple users.
- A set of findings and implications from users using finger orientation for touch discrimination purpose.

6.2 FUTURE WORK

The most exciting prospect for future work in this area is to increase the accuracy of using finger orientation to associate a touch point with its correct owner. Some newer techniques look promising; for example, Annett et al.'s [4] work, Medusa, which relies on proximity sensors to sense a user's standing position. Microsoft's new

surface [13] shows strong capability to capture accurate finger orientation more reliably. Dang and Andre's [5] FO algorithm can support multiple fingers. The research presented in my thesis opens up a number of other possibilities for future exploration:

User position SMESY does not directly identify users and cannot detect movement. The Position Avatar provides a basic and easy mechanism that allows users move around the table. More sophisticated methods include using overhead cameras, outward-facing infrared range sensors, or on-ground pressure sensor, which would limit the lightweight nature of my system.

Position profiles I collected FO profiles for specific positions around the table. This may suffice for many applications, but the fullest potential lies with fewer restrictions. It should be possible to generalize my approach to accommodate untrained profiles, for example a user standing at a corner. However, additional hardware might be required to track a user's position and orientation.

Number of users My studies investigated situations with up to three users. I believe that my system is extensible to more users without modification because I have explored the most difficult situations, side-by-side, and adjacent positions. Although my studies show adding more adjacent users might potentially decrease the SVM prediction accuracy, I do not expect performance to drop with users in a different side of the tabletop. However, additional investigation is necessary to scale my approach to a larger surface and to more users.

Multiple fingers I collected profiles for the index finger only. My system can be extended using existing algorithms (e.g. [5]) to detect multiple fingers from the same hand. However, no participant complained about the use of the index finger or found it limiting. I feel this restriction is not highly detrimental, given the benefits.

FO algorithm Most errors stemmed from my finger orientation algorithm. I expect that future systems will have bullet-proof methods for capturing finger orientation. Furthermore, secondary biometrics such as finger pressure could be leveraged to increase the accuracy of my system close to 100%.

Impersonation PAC is a valuable tool for error recovery, but could assist mischievous users in impersonating others. In most group situations, however, there is nothing to gain by impersonation. Also, social protocols, such as courtesy, or fear of being rejected by the group, might mitigate such issues. Future study outside a lab environment would provide further insight on this matter.

Target smaller devices My studies show one user per side achieve very high reliability. Testing with multiple users using a smaller device in the wild will be valuable to determine how well FO as a lightweight technique works in platforms other than tabletops. In addition, smaller devices, such as multi-touch tablets or even smartphones, are more mobile compared to tabletops. A lightweight touch discrimination technique will be highly desired.

Using finger only, in general, is the best way to perform tasks on touchable devices. Therefore, finger orientation is a natural attribute that designers can make use of to discriminate user touches. Although improvements to my technique will be necessary for users to accept a system using SEE ME SEE YOU in the wild, in all exit surveys, most participants responded positively when asked if the system is accurate enough for real-world use. As more advanced technology come to multi-touch devices, we will expect them to have better support for multi-user applications. Associating a user touch with its proper owner will only become increasingly important. My work, at the first time, explores, evaluates, and proves the feasibility of using finger orientation only to achieve touch discrimination and I hope research in this area continues.

BIBLIOGRAPHY

[1] Community core version. http://ccv.nuigroup.com/. [Online; accessed 18-May-2011]. (Cited on pages 7, 22, 26, and 28.)

[2] Phidgets 3052 relay board. http://www.phidgets.com/products.php?product_id=3052. [Online; accessed 24-Apr-2011]. (Cited on pages 22 and 27.)

[3] openframeworks. http://www.openframeworks.cc/. [Online; accessed 18-May-2011]. (Cited on page 27.)

[4] Michelle Annett, Tovi Grossman, Daniel Wigdor, and George Fitzmaurice. Medusa: a proximity-aware multi-touch tabletop. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, pages 337–346, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0716-1. (Cited on pages iii, vii, 14, 20, 66, and 70.)

[5] Chi Dang and Elisabeth Andre. Usage and recognition of finger orientation for multi-touch tabletop interaction. In *Human-Computer Interaction - INTERACT 2011*, volume 6948 of *Lecture Notes in Computer Science*, pages 409–426. Springer Berlin / Heidelberg, 2011. ISBN 978-3-642-23764-5. (Cited on pages 17, 23, 71, and 72.)

[6] Chi Tai Dang, Martin Straub, and Elisabeth André. Hand distinction for multi-touch tabletop interaction. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ITS '09, pages 101–108, 2009. ISBN 978-1-60558-733-2. (Cited on pages 13, 20, and 66.)

[7] Paul Dietz and Darren Leigh. Diamondtouch: A multi-user touch technology. In *UIST '01: Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*, pages 219–226, Orlando, Florida, 2001. ACM. ISBN 1-58113-438-X. (Cited on pages iii, vii, 2, 4, 7, and 9.)

[8] K.C. Dohse, T. Dohse, J.D. Still, and D.J. Parkhurst. Enhancing multi-user interaction with multi-touch tabletop displays using hand tracking. In *Advances in Computer-Human Interaction, 2008 First International Conference*, number 10–15, pages 297–302, Sainte Luce, Feb 2008. (Cited on pages 13 and 66.)

[9] George W. Fitzmaurice, Hiroshi Ishii, and William A. S. Buxton. Bricks: Laying the foundations for graspable user interfaces. In *CHI '95: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 442–449, Denver, Colorado, United States, 1995. ACM Press/Addison-Wesley Publishing Co. ISBN 0-201-84705-1. (Cited on page 4.)

[10] Jefferson Y. Han. Low-cost multi-touch sensing through frustrated total internal reflection. In *Proceedings of the 18th annual ACM symposium on User interface software and technology*, UIST '05, pages 115–118, 2005. ISBN 1-59593-271-2. (Cited on pages 6 and 21.)

[11] Christian Holz and Patrick Baudisch. The generalized perceived input point model and how to double touch accuracy by extracting fingerprints. In *Proceedings of the 28th international conference on Human factors in computing systems*, CHI '10, pages 581–590. ACM, 2010. ISBN 978-1-60558-929-9. (Cited on pages iii, vii, 11, and 12.)

[12] Andreas Holzammer. Combining diffuse illumination and frustrated total internal reflection for touch detection. Master's thesis, Technology University of Berlin, 2009. (Cited on pages iii, viii, 6, 20, 25, and 27.)

[13] Microsoft Inc. Experience this in a whole new way. `http://www.microsoft.com/surface/en/us/whatissurface.aspx`. [Online; Last accessed: November 2011]. (Cited on pages iii, vii, 7, 8, and 71.)

[14] Kenrick Kin, Maneesh Agrawala, and Tony DeRose. Determining the benefits of direct-touch, bimanual, and multifinger input on a multitouch workstation. In *Proceedings of Graphics Interface 2009*, GI '09, pages 119–124, Toronto, Ont., Canada, Canada, 2009. Canadian Information Processing Society. ISBN 978-1-56881-470-4. (Cited on page 24.)

[15] Nicolai Marquardt, Johannes Kiemer, and Saul Greenberg. What caused that touch?: expressive interaction with a surface through fiduciary-tagged gloves. In *ACM International Conference on Interactive Tabletops and Surfaces*, ITS '10, pages 139–142. ACM, 2010. ISBN 978-1-4503-0399-6. (Cited on pages iii, vii, 10, and 11.)

[16] Tobias Meyer and Dominik Schmidt. Idwristbands: Ir-based user identification on multi-touch surfaces. In *ACM International*

Conference on Interactive Tabletops and Surfaces, ITS '10, pages 277–278, 2010. ISBN 978-1-4503-0399-6. (Cited on pages iii, vii, 10, and 11.)

[17] Meredith Ringel Morris, Andreas Paepcke, Terry Winograd, and Jeannie Stamberger. Teamtag: exploring centralized versus replicated controls for co-located tabletop groupware. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, CHI '06, pages 1273–1282, 2006. ISBN 1-59593-372-7. (Cited on page 10.)

[18] N-trig. N-trig. http://www.n-trig.com/. [Online; Last accessed: November 2011]. (Cited on page 7.)

[19] William Newman and Pierre Wellner. A desk supporting computer-based interaction with paper documents. In *CHI '92: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 587–592, Monterey, California, United States, 1992. ACM. ISBN 0-89791-513-5. (Cited on page 4.)

[20] Alex Olwal and Andrew D. Wilson. SurfaceFusion: Unobtrusive tracking of everyday objects in tangible user interfaces. In *GI '08: Proceedings of Graphics Interface 2008*, pages 235–242, Windsor, Ontario, Canada, 2008. Canadian Information Processing Society. ISBN 978-1-56881-423-0. (Cited on page 5.)

[21] Grant Partridge. Identity awareness on tabletop computers. Master's thesis, University of Manitoba, 2011. (Cited on pages 8, 9, and 20.)

[22] Grant A. Partridge and Pourang P. Irani. Identtop: a flexible platform for exploring identity-enabled surfaces. In *Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, CHI EA '09, pages 4411–4416, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-247-4. (Cited on page 1.)

[23] James Patten, Hiroshi Ishii, Jim Hines, and Gian Pangaro. Sensetable: A wireless object tracking platform for tangible user interfaces. In *CHI '01: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 253–260, Seattle, Washington, United States, 2001. ACM. ISBN 1-58113-327-8. (Cited on page 4.)

[24] Anne Marie Piper, Eileen O'Brien, Meredith Ringel Morris, and Terry Winograd. Sides: a cooperative tabletop computer game for social skills development. In *Proceedings of the 2006 20th*

anniversary conference on Computer supported cooperative work, CSCW '06, pages 1–10, 2006. ISBN 1-59593-249-6. (Cited on page 10.)

[25] PQLabs. Multi-touch g3 touch screen. http://multi-touch-screen.com/product.html. [Online; Last accessed: November 2011]. (Cited on page 7.)

[26] Tim Roth. Multitouch dev blog. http://iad.projects.zhdk.ch/multitouch/?p=90. [Online; accessed 18-May-2011]. (Cited on pages iii, vii, 5, and 6.)

[27] Volker Roth, Philipp Schmidt, and Benjamin Güldenring. The ir ring: authenticating users' touches on a multi-touch display. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, UIST '10, pages 259–262, 2010. ISBN 978-1-4503-0271-5. (Cited on pages 2 and 10.)

[28] K. Ryall, A. Esenther, C. Forlines, C. Shen, S. Shipman, M.R. Morris, K. Everitt, and F.D. Vernier. Identity-differentiating widgets for multiuser interactive surfaces. *Computer Graphics and Applications, IEEE*, 26(5):56 –64, sept.-oct. 2006. (Cited on page 8.)

[29] Dominik Schmidt. Know thy toucher. In *CHI 2009 Workshop*, pages 4–9, Boston, MA, USA, 2009. (Cited on page 8.)

[30] Dominik Schmidt, Ming Ki Chong, and Hans Gellersen. Handsdown: hand-contour-based user identification for interactive surfaces. In *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*, NordiCHI '10, pages 432–441, 2010. ISBN 978-1-60558-934-3. (Cited on page 12.)

[31] Dominik Schmidt, Ming Ki Chong, and Hans Gellersen. Idlenses: dynamic personal areas on shared surfaces. In *ACM International Conference on Interactive Tabletops and Surfaces*, ITS '10, pages 131–134, 2010. ISBN 978-1-4503-0399-6. (Cited on pages vii, 1, 2, 12, and 13.)

[32] Stacey D. Scott. Territory-based interaction techniques for tabletop collaboration, 2003. (Cited on page 1.)

[33] Chia Shen, Frédéric D. Vernier, Clifton Forlines, and Meredith Ringel. Diamondspin: an extensible toolkit for around-the-table interaction. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '04, pages 167–174, 2004. ISBN 1-58113-702-8. (Cited on page 10.)

[34] Inc. Smart Technologies. Smart technologies, inc. http://www.smart-technologies.com/dvit.html. [Online; Last accessed: November 2011]. (Cited on page 7.)

[35] DISPLAX Interactive Systems. Displax products. http://www.displax.com/en/products/overlay-multitouch.html. [Online; Last accessed: November 2011]. (Cited on page 7.)

[36] Benjamin Walther-Franks, Marc Herrlich, Markus Aust, and Rainer Malaka. Left and right hand distinction for multi-touch displays. In Lutz Dickmann, Gerald Volkmann, Rainer Malaka, Susanne Boll, Antonio KrÃ¼ger, and Patrick Olivier, editors, *Smart Graphics*, volume 6815 of *Lecture Notes in Computer Science*, pages 155–158. Springer Berlin / Heidelberg, 2011. ISBN 978-3-642-22570-3. (Cited on page 25.)

[37] Feng Wang and Xiangshi Ren. Empirical evaluation for finger input properties in multi-touch interaction. In *Proceedings of the 27th international conference on Human factors in computing systems*, CHI '09, pages 1063–1072, 2009. ISBN 978-1-60558-246-7. (Cited on pages 15, 16, and 20.)

COLOPHON

This thesis was typeset with the `pdflatex` LaTeX 2_ε interpreter using Hermann Zapf's *Palatino* type face for text and math and *Euler* for chapter numbers. The listings were set in *Bera Mono*.

The typographic style of the thesis was based on André Miede's wonderful `classicthesis` LaTeX style available from CTAN. My modifications were limited to those required to satisfy the constraints imposed by my university, mainly 12pt font on letter-size paper with extra leading. Miede's original style was inspired by Robert Bringhurst's classic *The Elements of Typographic Style* [**?**]. I hope my naïve, yet carefully considered changes are consistent with Miede's original intentions.

Final Version as of March 1, 2012 at 9:52.