AN EXPLORATION OF MULTI-FINGER INTERACTION ON MULTI-TOUCH SURFACES

by

Shahzad Malik

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

Abstract

An Exploration of Multi-finger Interaction on Multi-touch Surfaces

Shahzad Malik
Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto
2007

Recent advances in touch sensing technologies have made it possible to interact with computers in a *device-free* manner, allowing for arguably more natural and intuitive input using multiple hands and fingers. Unfortunately, existing multi-point touch-sensitive devices have a number of sensor limitations which restrict the types of manipulations that can be performed. Additionally, while many well-studied techniques from the bimanual interaction literature are applicable to these emerging multi-point devices, there remain many unanswered questions as to how multiple fingers from a single hand can best be utilized on these touch-sensitive surfaces. This dissertation attempts to address some of these open issues.

We first develop the Visual Touchpad, a low-cost vision-based input device that allows for detecting multiple hands and fingertips over a constrained planar surface. Unlike existing multi-point devices, the Visual Touchpad extracts a reliable 2D image of the entire hand that can be used to extract more detailed information about the fingers such as labels, orientation, and hover. We then design and implement three systems that leverage the capabilities of the Visual Touchpad to explore how multiple fingers could be used in real-world interface scenarios. Next we propose and experimentally validate a fluid interaction style that uses the thumb and index finger of a single hand in an asymmetric-dependent manner to control *bi-*

digit widgets, where the index finger performs the primary and more frequent 2D tasks and the thumb performs secondary and less frequent tasks to support the index finger's manipulations. We then investigate the impact of visual feedback on the perception of finger span when using bi-digit widgets to merge command selection and direct manipulation. Results suggest that users are capable of selecting from up to 4 discrete commands with the thumb without any visual feedback, which allows us to design a set of more advanced bi-digit widgets that facilitate smooth transitioning from novice to expert usage.

Acknowledgments

Over the past five years I have had the privilege of working with a number of people that have made the usually long and exhausting PhD experience both intellectually stimulating as well as enjoyable. Each of these people deserves a huge thank you since they have all contributed to this dissertation in a variety of ways.

I am especially grateful to my advisor Allan Jepson for allowing me the freedom to pursue my research interests, even after my dissertation began to focus more towards Human-Computer Interaction and less on Computer Vision. Without this freedom and unwavering support, the work presented in this dissertation would not have been possible. His timely and critical evaluations of my work also lead me to believe that he should be on more HCI thesis committees! Also, by having Allan as my advisor, I can now count names like Euler, Lagrange, and Bernoulli as part of my PhD genealogy. How cool is that?

I would also like to thank all of the members of my thesis committee. First, I'd like to thank Ravin for unknowingly helping with my transformation from a vision student into an HCI student by first providing me with card access to the DGP lab when I took his excellent graduate HCI course and then, through some sort of osmosis process when I worked on calibrating the large display, giving me an awesome PC and permanent desk space in one of the most exclusive parts of the lab. Karan Singh also deserves a special thanks for providing the necessary connections and vision to make the Deaf Culture Centre interactive project a reality, which subsequently became an important part of this thesis. I'd also like to thank Ken Hinckley at Microsoft Research for agreeing to be the external appraiser for my final oral defense despite his busy schedule, particularly with the new twins at home. His deep insights and feedback regarding my research definitely made my final presentation much stronger.

Finally, Khai deserves a huge thank you for agreeing to be on my committee at the last second and for always asking the best questions after my presentations.

I would like to thank the Natural Sciences and Engineering Research Council of Canada (NSERC), the Ontario Graduate Scholarship program (OGS), and the University of Toronto for generous funding throughout my graduate studies. Without this financial support, my decision to attend graduate school would have been much more difficult. I would also like to thank Approach Infinity Media, 3DSolar, the Deaf Culture Centre, Google AdSense, the supporters and cheerleaders of CognoVision, and the fans of Extended Reality for an additional stream of funds throughout my studies. Of course, these side projects also made finishing the PhD much more difficult!

I would also like to thank everyone in the DGP lab for making it such an awesome place to work as well as have fun. In particular, I'd like to thank my desi-bro Anand Agarawala for introducing me to Kom Jug Yuen and for permanently embedding the term "crispy" into the DGP lexicon; Gerry Chu for the most entertaining and force-feedback inducing sneezes; John Hancock for making the lab equipment so easily accessible and running smoothly and for helping to make DGP social events so entertaining; Joe Laszlo for being a constant source of wacky research ideas, for teaching me how to do poi, and for simply always being there; Noah Lockwood for sharing the philosophy that the best thesis is the one that's finished; Nigel Morris for the regularly inconsistent workouts at Hart House; desi-bro Abhishek Ranjan for the awesome squash matches and our self-titled "kick-your-own-ass" circuit training sessions; and Jack Wang for the always entertaining discussions about life, politics, and the sick and evil society we live in. I'd also like to thank my fellow lab mates Anastasia Bezerianos, Xiaojun Bi, Xiang Cao, Pierre Dragicevic, Tomer

Moscovich, Gonzalo Ramos, and Daniel Vogel for the random but enjoyable discussions that we'd somehow get into as we worked away on our own research.

I would also like to thank the many interesting personalities that I came across during my time in Toronto, such as the Babi at Tim Horton's, Turrets vala Baba, Shame-on-You Guy, Zanta, Maple Leafs fans, the patrons of Brass Rail, and the laal-desi couple for confirming that the suburbs of Ottawa are indeed a more sane and peaceful place to live and raise a family.

Speaking of family, my parents deserve a huge thanks for their constant love, support, and patience throughout my studies. Only by instilling in me the belief that a solid education and strong work ethic are the keys to success was I able to come this far up the academic ladder. I'd also like to thank my sister Farina for always making my visits back to Ottawa so fun, particularly with the homemade feasts of pesto linguine, haleem, and Greek food while watching Senators games. The Sens jersey hanging in the window of her cozy living room was also always a nice touch. My sister Nadia, easily the best stylist in the entire city of Ottawa, also deserves a special thanks for all the free past and future haircuts, and for trying to teach me that there is life away from the computer (aik duniya, aik zindagee). Of course, the only way to reach her these days is through Facebook.

I would also like to thank my uncle, Dr. Mukhtar Malik, who unfortunately passed away late last year after a difficult battle with cancer and wasn't able to see me complete this thesis. He was the one who first convinced me to pursue the PhD, so I am certain that right now he is watching from above, with a huge smile on his face, knowing that the Dr. Malik name will continue for at least one more generation.

Finally, I would like to thank my wife, Jawairia, for her never-ending love, support, and encouragement throughout this PhD adventure. This thesis is as much hers as it is mine, and I can't thank her enough for putting up with the long hours I spent at the lab or sitting in front of the computer. For this reason, I dedicate this thesis to her.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Recent advances in touch sensing technologies have made it possible to interact with computers in a *device-free* manner, allowing for arguably more natural and intuitive input using hands and fingers. Such touch-sensitive surfaces now appear in many consumer electronics products ranging from laptops and PDAs to large electronic whiteboards and interactive tabletops. Unfortunately, most of these interactive surfaces only allow for very limited degrees-of-freedom that do not fully leverage the high bandwidth input capabilities of our hands. For example, most laptop touchpads, electronic whiteboards, and touch-sensitive kiosks only detect a single point of contact, allowing for only two translational degrees-of-freedom and a binary touch state. As a result, the user interfaces for such devices do not usually go beyond the standard Windows/Icons/Menus/Pointing Devices (WIMP) paradigm that is commonly found on our traditional desktop machines.

A few recent technologies have shown that multiple points of contact can be detected on these touch-sensitive surfaces, which opens the door to much more expressive interactions. For example, the latest Apple Powerbooks [Appl05] feature a touch-sensitive surface that can determine whether one or two fingers are making contact, which can be used to toggle between two different interaction modes. Some recent SmartBoards [Smar05] also allow two fingers to be detected to simulate a right mouse button while interacting with an upright

touch-sensitive display. Other devices are capable of detecting full hand postures and gestures [Fing05], as well as multiple hands or users [Tact05, Diet01]. Despite these features, however, there are still a number of technical limitations from a hardware perspective. In particular, these devices have difficulty disambiguating contact points, which makes it difficult to assign distinct roles to each hand or finger. Additionally, these devices are unable to determine accurate fingertip positions when other parts of the hand (such as the palm) are placed on the surface, which potentially forces a user to make uncomfortable postures to perform manipulations.

From an interaction perspective, there are also a number of open issues. While the utility of multi-point devices has commonly been demonstrated by using the index fingers of two hands for controlling bimanual interfaces [Wu03, Han05, Igar05, Wils05, Benk06], there has been very little investigation into how multiple fingers from a single hand can be used effectively. Such unimanual multi-finger techniques could be beneficial when interacting with touch-sensitive surfaces by enhancing existing single-finger techniques. For example, by using two or more fingers from the same hand, multiple parameters could potentially be manipulated simultaneously which may allow for more fluid operations that traditionally require a user to make explicit mode switches. While multiple parameters can also be manipulated using two hands, the potentially high bandwidth capabilities of multiple fingers from each hand can be used to complement existing bimanual techniques. Finally, single-handed multi-finger techniques may be desirable over bimanual techniques when working with portable handheld devices, since one hand is typically dedicated to holding the device which limits a user's ability to perform two-handed manipulations.

1.2 Contributions

In this thesis, we explore some of the open issues in multi-finger interaction, both from an input device perspective as well as an interaction and human factors perspective. We investigate how multiple fingers from a single hand can be used to control two or more parameters simultaneously on touch-sensitive devices, which enables manipulations that are arguably more fluid than the status quo single-point interaction techniques. The specific contributions of this thesis are as follows:

- We present the design and implementation of the Visual Touchpad [Mali04], a low-cost vision-based input device that pushes the capabilities of multi-point touch-sensitive surfaces by detecting finger information that is difficult to extract with existing multi-point devices (such as hand and fingertip labels, finger orientation, and continuous hover). While the accuracy of these output parameters is still quite limited, no other existing multi-point device provides all of this information simultaneously.
- We design and implement a set of interaction techniques and systems that demonstrate the capabilities of the Visual Touchpad in real-world applications while also exploring the design space of multi-finger input. In particular, we show how the Visual Touchpad can be used as a low-cost, non-intrusive, and fluid input device for manipulating pictures [Mali04], for interacting with large displays from a distance [Mali05], and for interacting with a permanent public exhibit [Mali06].
- We propose and evaluate an asymmetric two-fingered interaction style for a single hand that uses the relative position of the thumb as a secondary control to support primary manipulations that are performed with the index finger [Mali07a]. This investigation helps to increase our understanding of how to effectively use two fingers in a user interface while also establishing two-fingered asymmetry as a viable and valuable method for high degree-of-freedom input. We also design a variety of general-purpose two-fingered widgets based on the proposed interaction style that allow for fluid secondary operations on multi-touch devices.
- We study the impact of visual feedback on the perception of finger span when an asymmetric finger mapping is used for merging command selection and direct manipulation [Mali07b]. In particular, we discover that users are capable of using finger span for selecting discrete targets in an eyes-free manner as long as there are at most four discrete zones. Based on the results of our investigation, we design more advanced widgets which support smooth transitioning from novice to expert usage by adjusting the amount of visual feedback presented to the user.

1.3 Thesis Overview

This thesis first introduces the current state of the art in multi-point input devices and interaction techniques in Chapter 2. Chapter 3 then explores how low-cost cameras and simple computer vision algorithms can be used to build a multi-point touch-sensitive device that allows for extracting a number of finger attributes that are difficult or impossible to detect on many existing multi-point devices. Chapter 4 then presents three system designs that explore the multi-finger interaction capabilities of the input device developed in Chapter 3. We design and develop: i) multi-finger techniques for manipulating pictures on a standard desktop PC; ii) bimanual and multi-finger techniques that facilitate interactions with a large upright display from a distance; and iii) a public art installation that uses multi-finger input for demonstrating the expressiveness of hand shapes and motion from the perspective of the deaf community. Chapter 5 then motivates the asymmetric interaction style that leverages the thumb and index finger of a single hand for fluid manipulations on touch-sensitive devices. We develop a taxonomy of bi-digital tasks, validate the asymmetric style with a formal experiment, and present a variety of continuous and discrete bi-digit widget designs that adhere to the proposed interaction style. In Chapter 6 we then evaluate the impact of visual feedback when discrete bi-digit widgets are used to merge command selection and direct manipulation into a single, fluid, one-handed operation. Finally, Chapter 7 summarizes the results of the thesis and presents avenues for future work.

Chapter 2

Related Work

2.1 Enabling Technologies

A number of existing input devices already leverage the capabilities of multiple fingers. For example, the standard two-button mouse is probably the most familiar and widely accepted multi-finger input device currently available. Laptops which feature a touchpad are also frequently used in a multi-finger manner, where the index finger typically controls the mouse cursor position with the touchpad while the thumb controls the buttons mounted below the touchpad. Other common multi-finger input devices include video game controllers, as well as prototype devices such as the PadMouse [Bala98] and TouchMouse [Hinc99] which combine standard mice with touch sensors.

Our primary interest, however, is in *device-free* multi-finger interactions which do not require any secondary components such as knobs, buttons, or joysticks. Instead, movements of the hands and fingers should be sensed directly on an interactive 2D surface such as an electronic whiteboard or Tablet PC to perform more expressive direct manipulations. In this section we describe some of the current technologies that allow such multi-finger interactions to occur.

2.1.1 Sensor-based Touch Surfaces

Touch-sensitive surfaces which do not require any secondary hand-held implements are now widely available in a number of everyday consumer devices such as laptops, personal digital assistants (PDAs), and cellular phones (Figure 2.1). For example, the Synaptics TouchPad [Syna05, Blas04], which is currently found in over 50% of today's laptops, allows users to position a mouse cursor, scroll windows, or simulate mouse button clicks in a graphical user interface using simple movements of a finger on the touchpad surface.

Figure 2.1 – The Synaptics Touchpad can be found in various devices [Syna05]: (left) Laptops; (middle) PDAs; (right) Cellular phones.

The touchpad itself is an array of conductive metal electrodes covered by a protective layer that acts as both an insulator as well as a smooth surface suitable for prolonged usage. When a finger (which itself is an electrical conductor) makes contact with the touchpad surface, the respective electric fields of the finger and the touchpad interact with one another to form capacitance. By measuring the amount of capacitance at each of the electrodes, the system can generate a 2D image of the hand above the surface which can be used to pinpoint the absolute location of the fingertip on the touchpad (currently with an accuracy of $1/1000^{th}$ of an inch). In addition to position, the touchpad can also estimate fingertip pressure and movement velocity, which allows for simulating a variety of standard mouse operations.

The major drawback with standard capacitance-based touchpads, however, is the difficulty in estimating multiple simultaneous fingertip positions, particularly on the small touchpad regions found in today's laptops. Nevertheless, the recent Apple Powerbooks [Appl05] feature a capacitance-based touchpad that can differentiate between one and two fingers, which opens the door to more sophisticated interactions. However, it is not clear whether

their technology can simply detect the centroid between two fingers, or whether the electrode array is fine enough to pinpoint the actual positions of each of the fingertips.

Lee et al. [Lee85] were one of the first to successfully demonstrate a full-size multi-point touch tablet using capacitance. Using a 64x32 matrix of active sensors combined with an interpolation scheme, their system was capable of accurately detecting multiple finger positions along with a small degree of contact information/pressure (based on the amount of spreading at the tip of a finger).

More recent multi-point touch tablets build upon the system developed by Lee et al., but with much higher resolution sensor grids. For example, Rekimoto's SmartSkin system [Reki02] features a thin and flexible mesh of transmitter and receiver electrodes which output a two-dimensional matrix of signal values for objects located less than 10cm above the surface. This signal can be processed by a pattern-matching algorithm in order to detect various hand and finger gestures, and with a dense enough grid of sensors the system can also determine the shapes of objects on or close to the surface. With an 8x9 grid of sensors, where each cell measures 10x10cm, the SmartSkin can detect fingertip positions with an accuracy of about 1cm. Unfortunately, since the SmartSkin relies on capacitance to determine the 2D signal, it is difficult to consistently disambiguate between different fingertips depending upon the pose of the hand. Additionally, the estimation of distance between the surface and a finger is imprecise. Nevertheless, the system is capable of coarse fingertip hover detection, which can be used for new types of interactions.

While the SmartSkin is a research prototype, the Fingerworks iGesturePad [Fing05] and Tactiva's TactaPad [Tact05] are two recent commercially-available capacitance-based touchpads that are also capable of recognizing multiple hands and fingers as they make contact with the surface.

The capacitance-based technology used in the above-mentioned touchpads may also be used to detect finger contact on screens. Since a 2D mesh of electrodes cannot be overlaid in front of the display, capacitance-based touch-screens instead place a set of row-sensing electrodes

and column-sensing electrodes along the edges of the display and connect them with a transparent screen overlay that is capable of storing electric charge. This 1D projection scanning approach allows for accurately detecting a single point of contact, but it does not work well for multiple fingertips when two or more fingers overlap the same row or column electrode. Additionally, it does not allow for detecting the amount of pressure being applied.

The DiamondTouch technology [Diet01] is one of the more popular capacitance-based surfaces that uses only row and column sensing electrodes, but it has the added ability to differentiate between multiple users (up to 4) using a special receiver that is attached to each user's chair. Therefore when a user makes contact with the touchpad, capacitance signals are sent through the user and into the receiver to determine which part of the surface was touched for that particular user.

Another common technique used to detect touch directly on a screen is to place a conductive and a resistive layer over top of a standard display. By holding the two transparent layers apart by spacers and then running an electrical current through them, any touch on the screen causes the two layers to make contact. By measuring the change in the electrical field at the touched location, special hardware and software can compute the screen coordinates where contact was made. Unfortunately, much like capacitance technology, resistive touch-screens also have difficulty interpreting multiple simultaneous contact points. Another common problem with resistive touch-screens is the reduction in the amount of light from the monitor (usually about 25%) that is transmitted through the layers.

2.1.2 Vision-based Touch Surfaces

With the continuous improvements in computer processing power, it is now feasible to track multiple hands and fingers in real-time using simple computer vision techniques. This is important for our purposes since vision techniques could potentially be used in place of the sensor-based touchpads and screens described in the previous section, while also affording additional capabilities that go beyond them.

One of the earliest demonstrations of using computer vision for interaction was Krueger's VIDEOPLACE system [Krue85], which used background subtraction and edge detection to extract a silhouette of a user's hand in order to interact with virtual imagery on a large display. For example, a single outstretched finger could be used to "fingerpaint" on a virtual on-screen canvas, while five outstretched fingers were used as an erase command to clear the entire canvas (Figure 2.2). Unfortunately, the VIDEOPLACE system relied on a single camera which restricted the interactions to two dimensions. The commercially available GestureTek technology [Gest05] builds upon Krueger's work by introducing an additional camera in order to triangulate 3D positions as well.

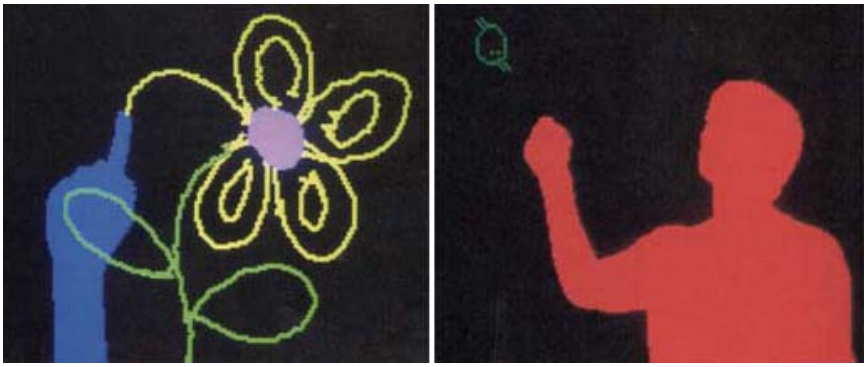


Figure 2.2 – VIDEOPLACE finger drawing application [Krue85].

The Visual Panel [Zhan01] uses computer vision to detect the tip of a finger over top of a white piece of cardboard, which effectively converts the cardboard into a wireless touch-sensitive surface (Figure 2.3). By mapping the corners of the cardboard to the corners of a display and then computing the corresponding fingertip position in screen space, the Visual Panel can be used to control the mouse cursor in a standard graphical user interface. Unfortunately, the Visual Panel does not detect multiple fingers and the single camera finger detector cannot differentiate between touch and hover states, which limits the types of interactions that can be performed.

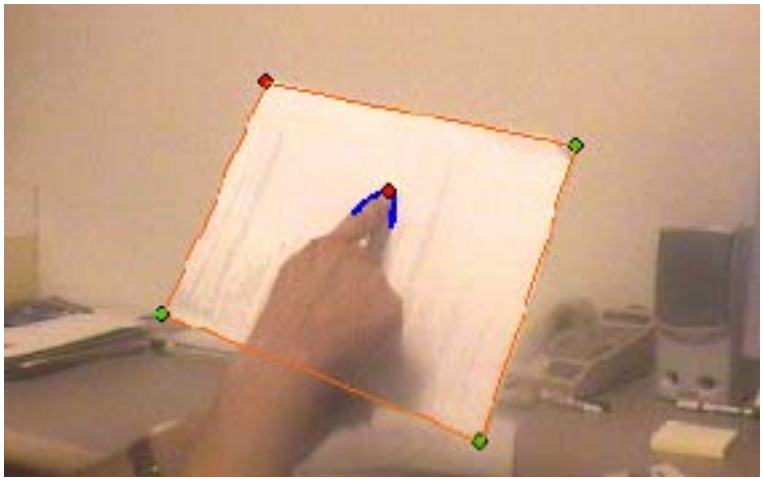


Figure 2.3 – Visual Panel [Zhan01].

In contrast, the GestureVR system [Sege98] demonstrated a stereo camera system that could perform real-time 3D hand tracking within a constrained volume around both cameras. Using a simple background subtraction scheme combined with a heuristic approach for determining the locations of the fingertips on the foreground hand silhouette, the system allowed an unmarked hand to be used to interact with virtual 3D objects.

Wellner's DigitalDesk system [Well93], which was one of the first top-projected interactive tabletops, also used computer vision to respond to bare finger interactions in a manner similar to the Visual Panel. Although the original DigitalDesk could only detect a single fingertip, the EnhancedDesk project [Oka02] demonstrated a multi-finger interactive tabletop that used an infrared camera for hand pixel detection and template matching for detecting fingertips. Much like the Visual Panel, however, their single-camera setup was incapable of extracting depth information for each fingertip which is useful for detecting actual contact with the tabletop surface.

Wilson's PlayAnywhere system [Wils05] is another interactive tabletop that uses computer vision techniques to detect hands and fingers on the surface. Although the system only uses a single infrared camera (along with an infrared illuminant), it is capable of detecting when the tip of a single outstretched finger touches the tabletop by analyzing the shape of the corresponding shadow (Figure 2.4). While this approach works well for simple single-finger interactions, the system cannot currently detect multiple fingertips for a single hand.

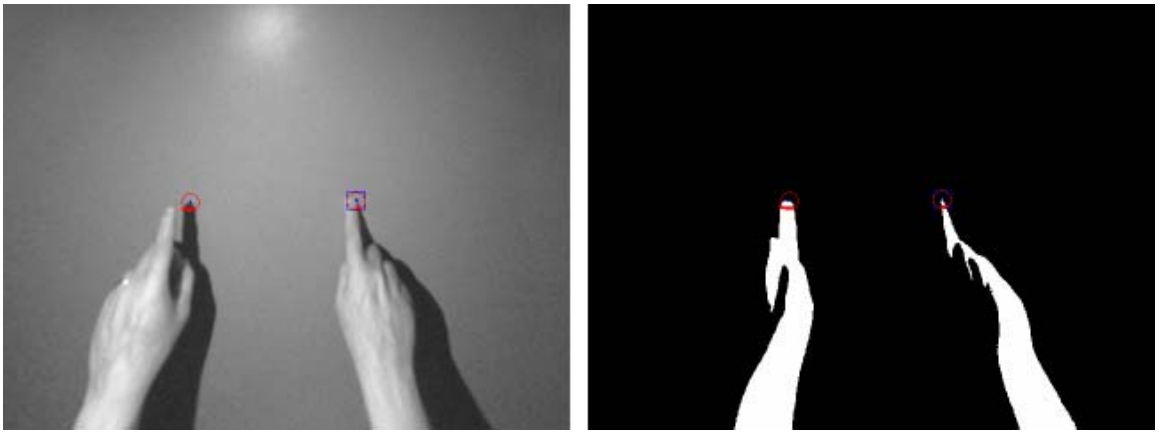


Figure 2.4 – Shadow shape analysis for contact detection [Wils05]. The shadow for the left index finger (which is hovering above the surface) is rounded, while the shadow for the right index finger (which is touching the surface) appears as a sharp sliver.

The HoloWall [Mats97] is a vision-based system capable of detecting multiple fingertips on an upright, rear-projected display. The system relies on a semi-opaque and diffusive sheet being placed behind the glass along with a projector, an infrared camera, and infrared light source. Objects that are more than 30cm from the glass screen are thus invisible to the camera (Figure 2.5) thereby allowing the system to detect contact points by simply thresholding the captured images. Wilson's TouchLight system [Wils04] builds upon the HoloWall system by introducing a second camera to estimate depth, which eliminates the need for a diffuser. This allows the cameras to capture objects beyond the interaction surface as well, such as faces for user identification or face-to-face interactive video conferencing.

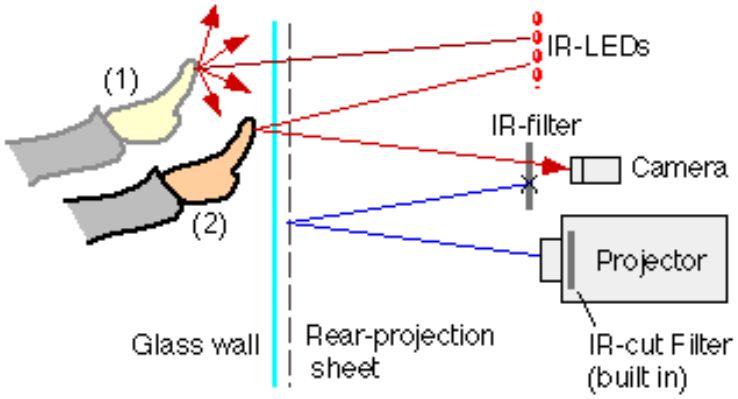


Figure 2.5 – Holowall contact detection [Mats97].

The system proposed by Han [Han05] leverages an optical property known as Frustrated Total Internal Reflection (FTIR) for detecting multiple fingers on a surface (Figure 2.6). As shown in the diagram, when light encounters a medium with a lower index of refraction (such as glass to air), it refracts by a certain amount based on the angle of incidence. Beyond a certain critical angle, the light undergoes total internal reflection which causes it to remain inside of the medium with the higher index of refraction. However, if some other material or object (such as a finger) interferes with the light at the interface between the two mediums, the light becomes *frustrated* and thus scatters through the other side of the medium as shown in the diagram. Therefore, by placing high-powered infrared LEDs along the edge of a piece of acrylic and then mounting an infrared camera behind it to capture the frustrated reflections, the surface effectively becomes a multi-finger touch-sensitive device. Simple image processing operations such as background subtraction and connected component analysis can then be used to extract the actual positions of a user's fingertips. The main drawback with the system, however, is the lack of a 2D image of the hand which is useful for such things as detecting hover, extracting finger orientation, and finger labeling/disambiguation.

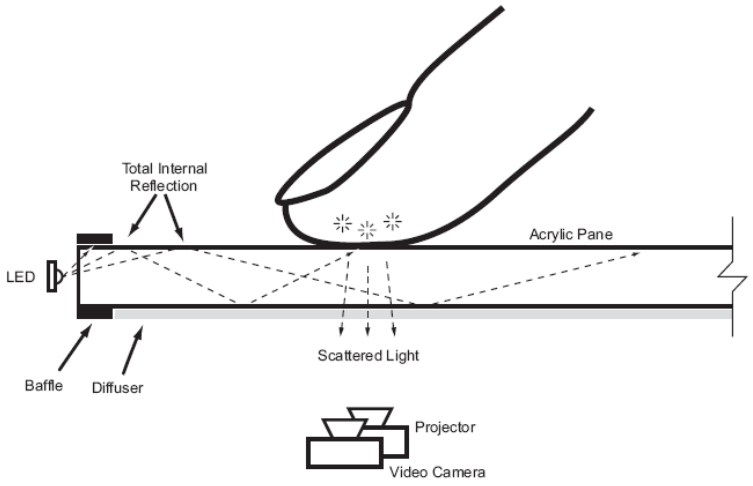


Figure 2.6 – Multi-finger surface using FTIR [Han05].

The commercially available DViT SmartBoards [Smart05] can also detect multiple fingers using computer vision techniques. By mounting cameras in either two or all four corners of a standard LCD, plasma, or rear-projected display, their systems can accurately triangulate up

to two points of contact on the surface. The technology can also detect a small amount of hover above the surface. Similar to the system by Han [Han05], however, the lack of a 2D touch image does not allow for labeling of fingers or extracting finger orientation.

While most of the vision-based multi-finger input technologies described thus far have attempted to extract depth maps or touch maps that can be used to determine the position of the fingertips, Corso et al. [Cors03] proposed an alternative approach that does not require any global image processing or finger tracking. Instead, graphical interface components such as buttons or icons each have their own region of interest (ROI) in the video stream. Therefore, each component simply monitors its ROI for a set of image and/or motion cues in a coarse-to-fine manner which correspond to certain actions. For example, a virtual button could first look for motion, followed by colour blob detection, finger shape verification, and finally disparity in order to be activated. While this approach is computationally efficient and works well for many familiar WIMP interfaces, it poses difficulties for applications where global hand information is important.

2.1.3 Glove-based Finger Tracking

Over the last two decades, virtual and augmented reality researchers have designed multi-finger input devices which use special gloves or markers for tracking hand motions in 3D space. Sturman [Stur94] provides a thorough survey of the various glove-based input technologies proposed in the virtual reality literature. In this section we will only briefly describe a few of the more recent glove-based input devices. Although our main focus is on device-free multi-finger interactions on planar surfaces, glove and marker-based input devices and interaction techniques are still relevant since the vision community is expected to eventually solve the markerless hand tracking problem.

Zimmerman et al. [Zimm87] presented the DataGlove, a glove-based hand tracking system that could measure the amount of finger flexion as well as the global 3D pose of the hand. The amount of finger flexion was determined using special sensors mounted inside the first layer of the two-layered nylon glove, while a Polhemus magnetic tracker attached to the dorsal side of the hand provided the pose. The DataGlove measured finger flexion angles

with an accuracy of about 5 to 10 degrees, but was incapable of measuring finger abduction (sideways movement).

The commercially available CyberGlove II [Imme05] is currently the most popular glove-based hand tracking solution (Figure 2.7). It has three flexion sensors per finger, four abduction sensors, and a palm-arch sensor. Combined with a single Polhemus magnetic tracker, the CyberGlove II can also provide a global 3D hand pose. Although the device does not directly provide 3D position information for each fingertip, they can be estimated using the global hand pose and finger flexion/abduction sensors.

Figure 2.7 – CyberGlove II device [Imme05].

A more affordable alternative to the CyberGlove for simple hand-based interactions is the Pinch Glove [Fake05], which is a cloth glove featuring electrical sensors located at each fingertip. Contact between any two or more fingertips completes a conductive path, allowing for a variety of "pinch" gestures to be programmed for different operations. Unlike the CyberGlove, however, the Pinch Glove alone cannot estimate finger flexion or position information.

Finally, Grossman et al. [Gros04] used a high-end optical tracking system that could determine the position and bending of a user's thumb and index finger. Their tracking system was free of any wires or electronics, and only required a user to wear small reflective markers on each finger as shown in Figure 2.8. Vogel and Balakrishnan [Voge05] also used a high-end optical tracking system for "Minority Report" style interactions by outfitting a regular pair of gloves with the same reflective markers.

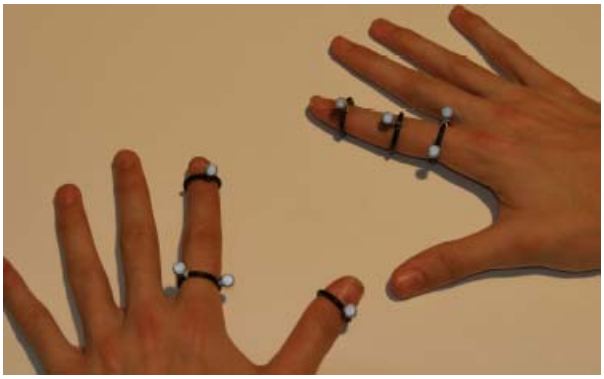


Figure 2.8 – Reflective markers for thumb and index finger tracking [Gros04].

2.2 Interaction Techniques

All existing multi-finger interaction techniques can be categorized into the following five major gesture categories as proposed by Karam and Schraefel [Kara05]:

Deictic: gestures which involve pointing in order to establish the identity or spatial location of a virtual object.
Manipulative: when the physical gesturing of the hand is tightly coupled with the manipulation of a virtual object [Shne83].
Semaphoric: static hand poses or dynamic hand motions which serve as commands to a system which is programmed to recognize them.
Gesticulation: natural hand gestures (as opposed to specific learned poses) combined with speech which define the context of the desired interaction [Rime91].
Language Gestures: gestures which are grammatically and lexically complete, and thus require more sophisticated recognition algorithms (similar to speech recognition) that go beyond semaphoric recognition systems.

Most interactive systems rarely use the five gesture styles in isolation, and instead combine two or more styles together. The majority of interaction techniques that are relevant to multi-point touch surfaces typically use a combination of deictic, manipulative, and semaphoric gesture styles.

Baudel and Beaudouin-Lafon [Baud93] suggested the following potential advantages of using hand gestures for input:

- Natural interaction: We already use gestures naturally for communication, and gestures are also easy to learn.
- Terse and powerful: A single gesture can be used for specifying both a command as well as any parameters.
- Direct interaction: The hand can be used directly for input, without the need for any secondary devices.

However, they also mentioned the following potential limitations:

- Fatigue: Long-term hand gesturing is difficult for extended periods of time.
- Lack of comfort: Tethered glove-based devices restrict autonomy and possibly even the range of motion.
- Non self-revealing: The user must know the set of gestures in advance.
- Immersion syndrome: The system should only interpret hand motions intended for it.

In the following sections we will describe the various multi-finger interaction techniques that have been proposed in the literature. The techniques are categorized based on the three most relevant gesture styles, and each system is assessed using Baudel and Beaudouin-Lafon's set of potential limitations.

2.2.1 Deictic Gesture Systems

Deictic gestures are usually implicit in systems which incorporate manipulative gestures [Kara05], since pointing usually precedes the actual manipulation of a virtual object. Nevertheless, exclusively deictic gestures can still be useful in certain applications. For example, the DigitalDesk calculator application [Well93] allowed a user to point to a number printed on a real paper document, which caused the system to recognize the digits and then automatically enter them into a virtual calculator. Although the DigitalDesk could only recognize a single index finger which is sufficient for simple pointing tasks, more recent interactive tabletops should be able to combine deictic gestures with semaphoric gestures so

that different outstretched fingers during pointing could represent different commands or operations on the targeted object.

2.2.2 Semaphoric Gesture Systems

In the Charade system [Baud93], Baudel and Beaudouin-Lafon used freehand semaphoric gestures with a DataGlove to control a slideshow presentation application (Figure 2.9). For example, a hand motion from left to right in front of an upright display represented the "next slide" command, while a right to left motion issued a "previous slide" command. In total, their system recognized 16 gestures, where each gesture varied based on the direction of hand motion and the number of fingers that were bent at the start, during, and end of the movement.

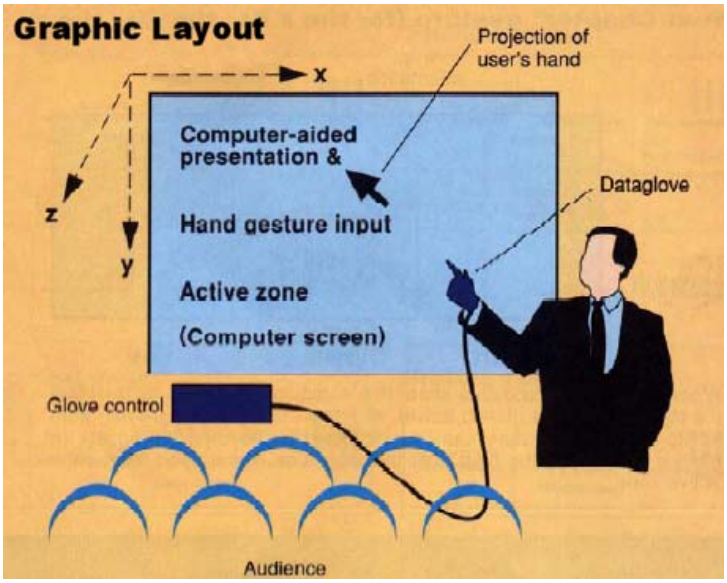


Figure 2.9 – The Charade system used semaphoric gestures for controlling a presentation [Baud 93].

In addition to a description of the Charade system, Baudel and Beaudouin-Lafon outlined a set of design guidelines when using semaphoric hand gestures for interaction. To deal with the issue of fatigue, they suggested using hand gestures that are very quick and concise, and to avoid gestures that require high precision movements. They also recommended using appropriate visual feedback to reduce the non self-revealing aspect of semaphoric gestural systems. Finally, to deal with immersion syndrome, their system defined an active zone in

front of the display so that only hand gestures inside of this zone would be interpreted by the system.

The GestureVR system by Segen and Kumar [Sege98] used semaphoric gestures for interacting with virtual 3D worlds. Using computer vision techniques, their system could recognize four distinct multi-finger gestures: a pointing gesture with the index finger, a pinching gesture where the thumb and index finger are outstretched, a clicking gesture where the index finger is quickly bent and then outstretched, and a reaching gesture where all five fingers are outstretched. All other gestures, including the absence of the hand, represented ground (no action).

With this simple gesture set, they demonstrated controlling a virtual fly-through of a 3D landscape. By placing one hand in the pinch pose and moving the hand forward or backward on the desk relative to a fixed "zero" position, the flying velocity could be increased or decreased. In this same pose, hand rotations controlled the yaw, pitch, and roll parameters of the camera. They also demonstrated this gesture set being used to control a first-person perspective 3D video game, where the pointing gesture was used to control the movement of the player, the clicking gesture was used to fire a weapon, and the reaching gesture was used as an action command (eg. to open doors).

Unfortunately, no user study was performed for their system, so it is difficult to determine how well their system works in practice. Nevertheless, the basic setup of their system appears to address some of the issues described by Baudel and Beaudouin-Lafon. For example, many of the gestures that have the potential to be used often are performed with the hand resting on the desk, which reduces fatigue. Additionally, using computer vision instead of a glove-based device potentially increases the comfort of the system. Finally, the small gesture set allows users to learn the system quickly, while at the same time reduces the effects of immersion syndrome.

The virtual and augmented reality communities have also investigated using glove-based input devices for multi-finger interactions. In addition to direct manipulations, Sturman et al.

suggested using a DataGlove as an abstracted graphical input device such as a button or valuator [Stur89]. For example, the bend sensors of the fingers were shown to be very effective when used to control the cursor position in a vertical menu. Their system also demonstrated a variety of simple temporal gestures for issuing single commands, such as twisting the wrist to simulate a button click or a fist posture to simulate clutching during cursor control.

The Responsive Workbench [Cutl97] combined two Pinch Gloves with Polhemus 6-DOF trackers to translate, rotate, and scale objects in an augmented 3D environment. An interesting observation that was made with the Responsive Workbench was that users often performed a number of unexpected bimanual manipulations using two otherwise independent single-handed tasks. In other words, these bimanual techniques weren't explicitly programmed by the developers, but instead emerged naturally.

The Flex and Pinch system [Lavi99] used customized Pinch Gloves with extra contact sensors placed strategically over the entire glove for advanced multi-finger selection operations in a 3D virtual world. For example, a user could select distant 3D objects by first aligning the desired target between the thumb and index finger, followed by contact between the index and middle fingers to complete the operation (left of Figure 2.10). Alternatively, distant objects could also be selected by casting a ray in the direction of the outstretched index finger followed by contact between the thumb and the side of the middle finger (right of Figure 2.10).

Figure 2.10 – Flex and Pinch system for selecting objects [Lavi99].

Vogel and Balakrishnan [Voge04] used a glove-based tracking system for semaphoric gestural interaction with large public displays from a distance. To enhance both recognition

as well as user performance, they relied on postures and temporal gestures that were nearly orthogonal to one another in terms of wrist and elbow angles (Figure 2.11). They also used the postures and gestures for both continuous and discrete operations. For example, the palm vertical gesture (Figure 2.11c) with left/right motion was used to control the position of a selection cursor in a horizontal menu, while flicking upwards acted as a binary "cancel" operation and flicking downwards locked the position of the selection cursor.

While the simple gesture set reduced the amount of training required to use the system effectively, Vogel and Balakrishnan also developed a self-revealing instructional video for new users that was activated based on a certain period of inactivity. Additionally, visual cues were presented for each posture to remind users of the various options that were available while also providing feedback that the recognition system was working correctly.

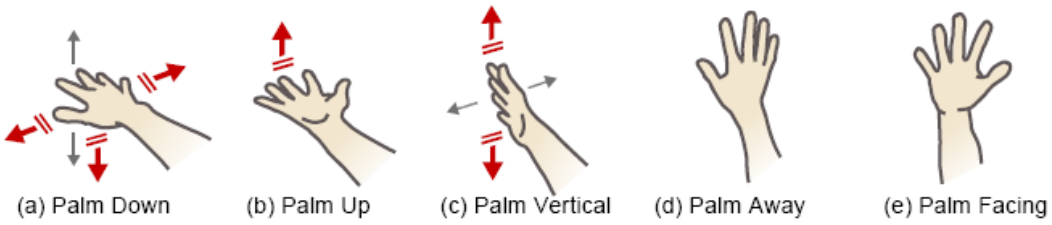


Figure 2.11 – Orthogonal postures and gestures ease both recognition and user performance (red arrows represent discrete gestures; grey arrows represent continuous gestures) [Voge04].

The recent work by Grossman et al. [Gros04] explored using semaphoric hand gestures for bimanual interactions with a hemispherical volumetric display. To simulate a touch-sensitive surface their system used the high-end marker-based tracking system described in Section 2.1. This allowed their system to recognize the following six postures and gestures: a pointing posture where the fingertip touches the display, a pointing posture where the index finger is parallel to the surface of the display, a pinch posture where the thumb and index finger are brought together, a curl posture where the index finger is bent, a trigger gesture where the thumb is pressed against the side of the index finger, and a scrub gesture where the thumb is rubbed along the side of the index finger. These gestures were put to use in a simple 3D model building application. For example, touching the surface of the display with the dominant hand's index finger followed by finger movement would cause an object to rotate while holding a pinch gesture followed by hand movement would cause an object to be

translated. Bimanual techniques were also explored, such as using two fingers to uniformly scale an object. The distance between the fingers was used as a parameter, so that spreading the fingers apart would increase the scale while moving the fingers closer together would decrease the scale.

By defining a "touch" state, a "hover" state, and a "no touch" state, Grossman et al. adhered to Baudel and Beaudouin-Lafon's design guideline regarding defining an active zone to reduce immersion syndrome. This design choice also closely follows Buxton et al.'s suggestion of using a three-state model for effective interactions with touch-sensitive surfaces [Buxt85]. Finally, although no formal user study was performed, it can be argued that by keeping the set of gestures both subtle and simple (only requiring two fingers), user fatigue was minimized.

The Barehands system [Ring01] uses a rear-mounted infrared camera to detect contact on a large upright rear-projected display similar to the HoloWall described earlier. In addition to supporting standard mouse operations with a single finger, their system also allows for the detection of various hand postures such as two fingers, the side of the hand, or a flat hand. These shapes are mapped to common commands such as copy and paste that normally must be selected from pull-down menus, thereby reducing the time required for interacting with the system.

Semaphoric gestures have also been demonstrated on multi-finger touch-sensitive tabletop displays. Rekimoto's SmartSkin system [Reki02], for example, could recognize when the palm of the hand was placed on the surface in order to activate a menu of options underneath the fingertips. Similarly, the system could recognize gestures where the thumb and index finger were either brought together or spread apart while touching the surface, with each gesture representing a "pick-up object" and "drop object" command respectively.

The advantage of using a touch-sensitive surface over freehand gestures is that the issue of immersion syndrome is automatically eliminated, since no action occurs when the hand is not touching the surface. However, to reduce fatigue it would be desirable to allow users to rest

their hands on the touch surface, so interaction techniques or recognition algorithms that can differentiate between an active and resting hand would be required [Kjel97].

2.2.3 Manipulative Gesture Systems

In addition to semaphoric gestures, Rekimoto also demonstrated manipulative gestures on the SmartSkin multi-finger device [Reki02]. In a map browsing application, for example, a user's fingers could act as virtual pins or anchors onto the corresponding positions of the map image. Therefore, the entire map could be panned in any direction by simply moving the fingers in the appropriate direction. Similarly, by changing the distance between the fingers the map could be zoomed in or out, while rotating the fingers allowed the entire map to be rotated (Figure 2.12). Rekimoto reports that users found this interaction style to be very intuitive, since it allowed the map to be manipulated in a manner similar to a real paper map.

Figure 2.12 – Multi-finger curve manipulation and map browsing [Reki02].

Wu and Balakrishnan [Wu03] also demonstrated a number of manipulative and semaphoric multi-finger gestures by combining a DiamondTouch surface with a top-projected display. For example, a user could double-tap on the surface with a single finger to activate a transparent tool palette, while a second finger could be used to make selections in a manner similar to the Toolglass approach proposed by Bier et al. [Bier93]. Other semaphoric techniques included one- and two-handed gestures for defining editing planes or displaying private information, while other manipulative gestures included a multi-finger freeform object rotation technique and a two-fingered continuous parameter adjustment widget.

Although they did not perform a formal evaluation of the system, informal user feedback showed that participants were able to learn the set of gestures with very little practice.

The work by Igarashi et al. [Igar05] demonstrated the intuitiveness of using multiple fingers and direct manipulation for animating non-rigid two-dimensional objects on a tabletop display. Similar to Rekimoto's map browsing application, Igarashi's system allowed the tips of a user's fingers to act as constraints on the underlying triangular mesh for an object. Therefore, by touching any part of an object and then sliding the fingertips, the object could be rotated, scaled, and deformed in a realistic manner (Figure 2.13).

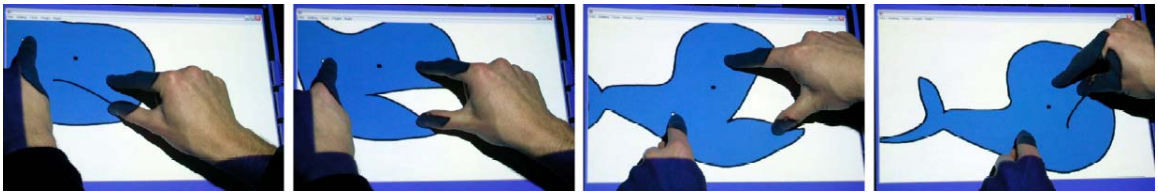


Figure 2.13 – Interactive 2D shape manipulation using multiple fingers [Igar05].

As mentioned earlier, Krueger's VIDEOPLACE [Krue85] was one of the first systems to demonstrate compelling direct finger manipulations with an upright display. In addition to standard single-finger operations such as selecting objects or finger painting, Krueger demonstrated a multi-finger curve manipulation application where the thumb and index finger of each hand could be used to simultaneously manipulate up to four control points of a Bezier curve (right of Figure 2.14). Since the set of actions in VIDEOPLACE mimic the way in which we interact with objects in the real world, the system requires little or no training. Unfortunately, the system suffers from immersion syndrome since there is no notion of an active zone as in the Charade system. For example, in the curve manipulation application, the system has no way (other than dwell time) to detect whether the user wants to release the control points. As a result, the curve will continue to "stick" to the user's hand as long as the thumb and index finger remain outstretched.

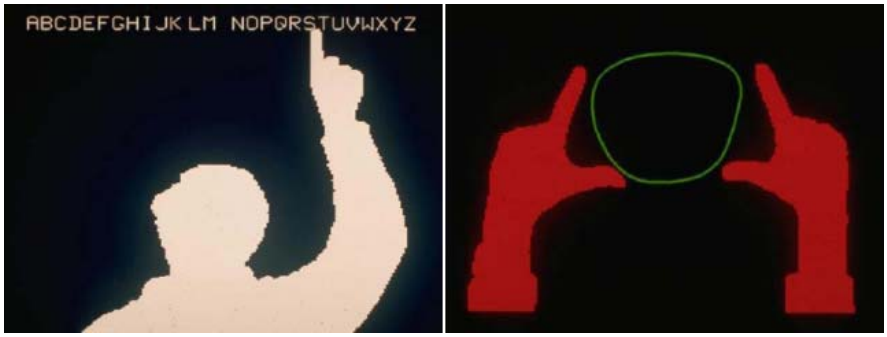


Figure 2.14 – VIDEOPLACE text entry (left) and curve manipulation (right).

A good example of using hand gestures for direct manipulation in an augmented environment is the FingARtips system by Buchmann et al. [Buch04]. Using a specially marked glove and vision-based tracking, their system could track the tips of the index finger and thumb, as well as the position of the joint between these two fingers. They implemented an urban planning application where users were able to grab and release virtual buildings by directly "pinching" objects with the thumb and index finger (Figure 2.15). Objects could also be moved or rotated based on the relative change in the axis joining the two fingers during a grab operation.

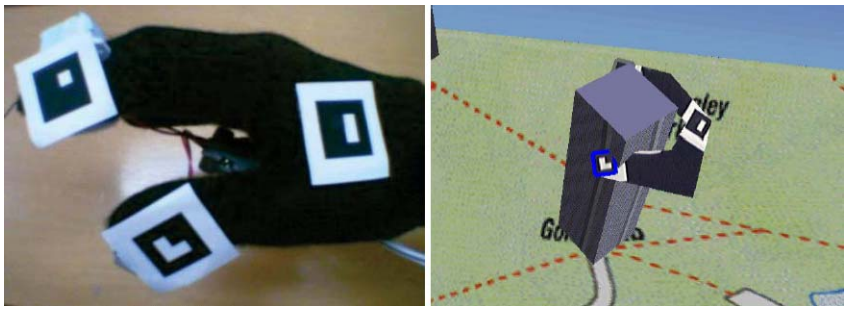


Figure 2.15 – The FingARtips system: (left) the glove and tracking markers; (right) the hand as it appears in the application [Buch04].

While a number of systems have demonstrated using the hand for directly controlling a cursor on a large display from a distance, Vogel and Balakrishnan were one of the few to leverage the capabilities of multi-finger tracking [Voge05]. For example, their ThumbTrigger selection technique allowed a user to simulate mouse clicks by simply tapping the thumb against the index finger. They also made use of finger joint angles with their AirTap selection technique, which allowed users to simulate mouse clicks using the same index finger that could be used for ray-casted cursor control.

Chapter 3

Visual Touchpad: A Multi-Finger Input Device

3.1 Introduction

As discussed in Chapter 2, existing multi-point touch-sensitive devices have a number of limitations that potentially restrict the types of interactions that can be performed on them. The first is the inability of many devices to disambiguate between two or more contact points, which make it difficult for an interaction technique to assign distinct roles to different hands and fingers. Another shortcoming with standard touch sensitive devices is that they usually only recognize hand gestures on or very close to the surface. Rekimoto's Smartskin technology [Reki02] can detect hand proximity to some extent, but it is still difficult to determine specific feature points for hand postures and gestures beyond a few centimetres above the surface. At the same time, however, existing devices have difficulty interpreting fingertip positions when other parts of the hand are contacting the surface as well. Therefore, a user often cannot place the palm of their hand on the surface, even if this increases comfort. Another problem with touch sensitive surfaces is the lack of robust finger orientation information, which is useful for certain types of operations. In other words, while accurate position information can be determined for the tip of a finger touching the surface, it is very difficult to determine in which direction the finger is pointing without requiring the whole finger to be placed flat on the surface.

In this chapter, we explore the idea of using computer vision techniques to track a user's bare, unmarked hands along a planar region called the Visual Touchpad that simulates a multi-point touch-sensitive surface. By using stereo vision we can not only determine contact information, but also a measure of a fingertip's height above this Visual Touchpad surface for additional types of input. We can also use vision techniques to extract finger labels and orientation information for other more advanced interactions. Such a device allows for direct two-handed and multi-finger gestural interactions on desktop PCs, laptops, or public kiosks.

3.2 System Overview

Our work largely builds upon the Visual Panel proposed by Zhang et al. [Zhan01] and described in Chapter 2. In their system they track a quadrangle shaped piece of paper using single-view computer vision techniques, and then extract the position of a fingertip over the panel in order to position the mouse cursor in a Windows desktop. Since the panel is not equipped with any buttons or touch sensors, mouse clicks are simulated by holding the fingertip position steady for one second. Text entry is achieved by way of a virtual on-screen keyboard. Due to the one second delay, text entry and interface navigation can be quite slow. Additionally, the single fingertip detector only allows for two degrees of freedom, thereby limiting the input to single cursor mouse control. However, by extracting the X and Y orientation of the actual panel from some base pose, they are able to simulate a joystick which is useful for another two degrees of freedom.

3.2.1 Hardware

Similar to the Visual Panel, the Visual Touchpad is a simple quadrangle panel such as a piece of paper with a rigid backing, over which hand gestures can be recognized for interaction purposes. In our system, we use a piece of paper with a large black rectangle in the centre, surrounded by a thin white border. This black region defines the active "touchpad", while the white border facilitates the vision algorithms described later. The touchpad can be any size as long as we can place both hands comfortably over top of it. Additionally, if the touchpad will be mapped to a display in an absolute manner, then the touchpad and the display should ideally have the same aspect ratio.

Two off-the-shelf web cameras are then placed in a convenient location such that the black rectangular region of the touchpad is fully visible to both cameras, and the cameras are placed with a sufficiently wide baseline for accurate depth estimation. The cameras can capture 320x240 images at 30 frames per second on a standard Pentium 4 PC. For desktops, laptops, and kiosk configurations, it is sufficient to fix the location of the touchpad in front of the display, and then place the cameras on top of the display facing downward (Figure 3.1a and Figure 3.1b). We imagine that hand-held configurations are also possible if the cameras can be mounted underneath a portable touch surface, but we have not yet fully explored this particular configuration.

(a) **(b)**

Figure 3.1 – Example configurations for the Visual Touchpad: (a) Desktop; (b) Laptop.

3.2.2 Homography Computation

To simulate a touch-sensitive tablet, we first compute the mapping between the touchpad in each camera's view into a common touchpad coordinate space (Figure 3.2). In order to determine this mapping, we use a homography [Faug01], which defines a plane-projective mapping between two planes. To compute a homography we require the positions of at least four points on one plane and the corresponding four points on the other plane.

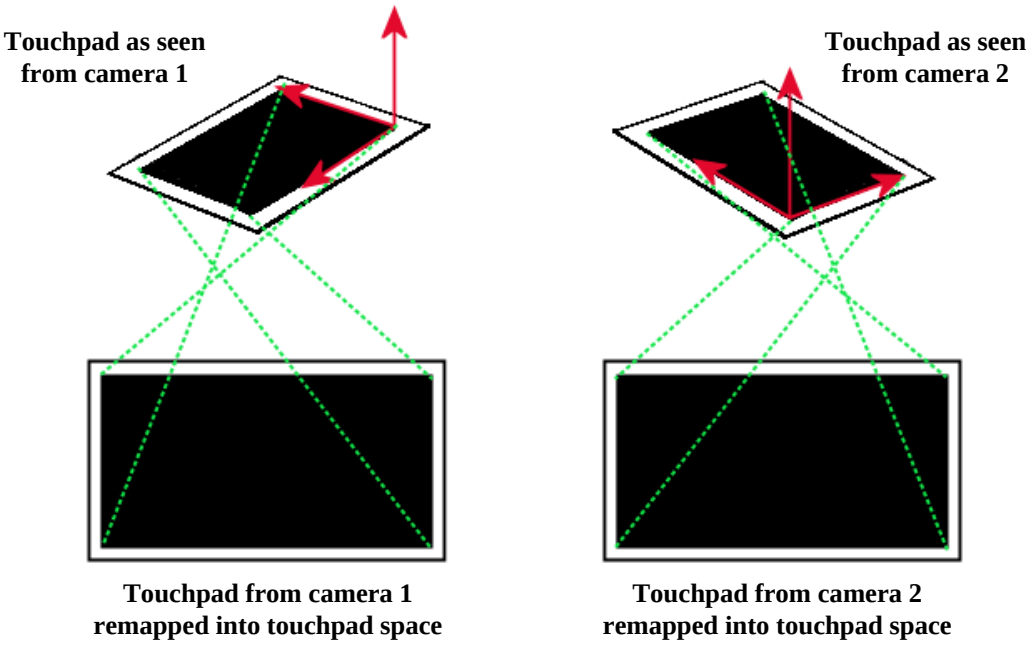
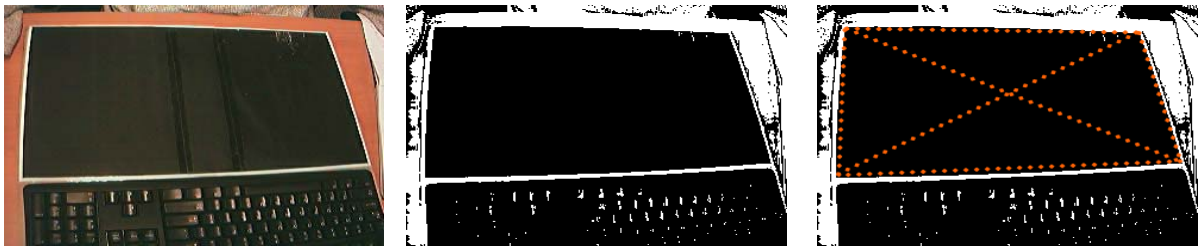


Figure 3.2 – Image space to touchpad space mapping.

Therefore, for each of our cameras, we detect the four corners of the touchpad in a captured video frame and then compute $H_{i \in \{1,2\}}$, which represents the 3x3 homography matrix that maps camera *i*'s view of the touchpad into touchpad coordinates. The corresponding metric coordinates of the touchpad can be determined by physically measuring the dimensions of the black region of the touchpad with a ruler or measuring tape (we currently use a touchpad of 472 x 274mm).

To find the corners of the touchpad in a frame of video, we use simple binary image processing operations. First we threshold a grayscale version of the video frame into a binary image in order to segment out the high contrast black rectangle that is surrounded by the thin white border (Figures 3.3a and 3.3b). We currently use a fixed value of 128 for our 8-bit grayscale image threshold, which works well in most situations. Connected component analysis is then performed to extract the largest black connected region in the video frame, and for this black blob we extract the four strongest corner features (Figure 3.3c). A homography is then computed using these four touchpad corners and the corresponding corners in touchpad space. Two main assumptions are made to detect the touchpad: 1) the largest black blob region will correspond to the black rectangular region of the touchpad; and

2) each camera views the touchpad so that the top-left, top-right, bottom-left, and bottom-right corners of the touchpad are in the bottom-right, bottom-left, top-right, and top-left quadrants of the video frame respectively.



(a) **(b)** **(c)**

Figure 3.3 – Touchpad detection: (a) Original frame; (b) Thresholded binary image; (c) Corners detected.

3.2.3 Hand Tracking

In this section we describe the details of the hand tracker, which applies low-level image processing operations to each frame of video in order to detect the locations of the fingertips. While a model-based approach that uses temporal information could provide more robustness to situations such as complex backgrounds or overlapping hands, the image processing approach is straightforward to implement and can run in real-time with low cost PCs and cameras.

3.2.3.1 Image Rectification

Using H_i defining the mapping from the touchpad in camera i to touchpad space, our hand tracker first warps each frame of live video so that the touchpad (and any hand over top of it) is in touchpad space. Let p_j represent a pixel in image space using homogeneous coordinates, and q_j represent the corresponding pixel from touchpad space in homogeneous coordinates. Therefore we have

$$q_j = H_i^{-1} p_j$$

Figure 3.4a and 3.4b show the result of creating a rectified (touchpad space) image of the touchpad with a hand over top of it.

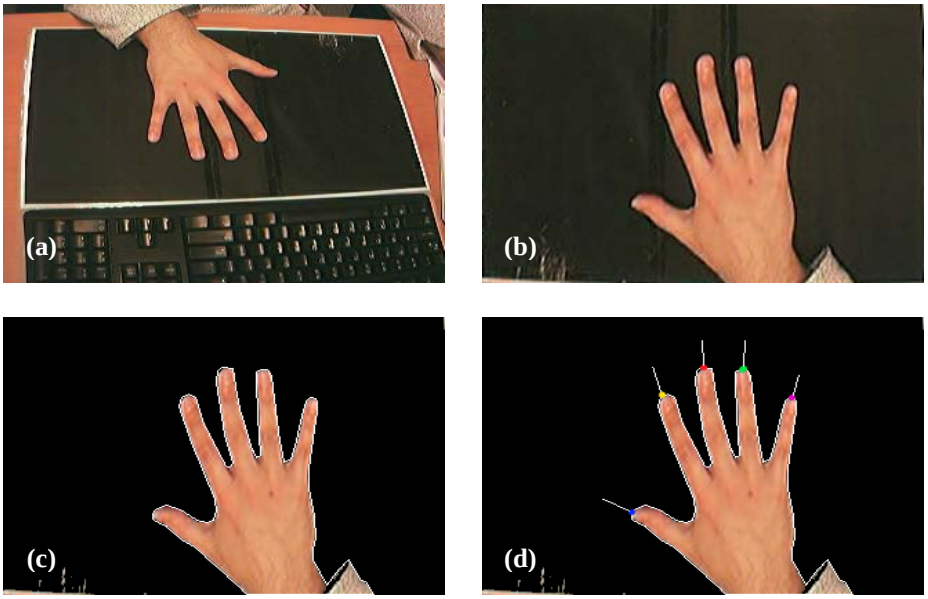


Figure 3.4 – Hand detection in the rectified image: (a) Original image; (b) Rectified image; (c) After background subtraction and contour detection; (d) Finger tip positions and orientations detected.

3.2.3.2 Background Subtraction

Since we assume a black rectangular region for our touch surface, it is easy to segment out the hand from the rectified image by using a simple background subtraction operation (Figure 3.4c). By using a black region as our known background, the system is robust to shadows cast onto the touchpad by foreground objects such as hands. Additionally, the system can reliably detect foreground objects in a wide variety of lighting conditions as long as they are different from the black background.

3.2.3.3 Hand Blob Detection

Connected component analysis is then performed in order to extract the foreground objects, and the two largest connected regions above some threshold size are assumed to be the hand blobs. Assuming that hands will not cross over during interaction, we simply label the left-most blob as the left hand, and the right-most blob as the right hand. In the case of only a single blob, we consider it to be either the left or right hand depending on a software setting that defines a user's dominant hand preference. For each detected hand, contour points are extracted in a clockwise order.

3.2.3.4 Fingertip Detection

Potential fingertips are found by finding strong peaks along the contours of each detected hand. We first use an approach similar to [Sege98], where the vectors from a contour point *k* to *k*+*n* and *k*-*n* are computed (Figure 3.5). We currently use a value of 16 for *n*. If the angle between these vectors is below some threshold (we currently use 30 degrees) then we mark that contour point as a potential fingertip. To avoid detecting valleys (such as between fingers) we verify that the determinant of the 2x2 matrix consisting of the two vectors is negative.

This process will result in a set of candidate peaks around the actual fingertips. Small protrusions along the contour (such as knuckles) will not be detected as peaks due to the chosen angle threshold and *n* value. Additionally, we remove potential peaks that occur within a threshold distance from the bottom and sides of the touchpad to avoid false positives that occur when the hand is partially clipped in the rectified image. To compute the final fingertip positions we search for the peaks that have the smallest angle among their +*n* and –*n* neighbours (non-maximal suppression). However, rather than using this local peak as the final fingertip position, we perform a weighted average of the +*n* and –*n* neighbouring contour positions, where the dot product of the peak angle is used as the weight. This results in a final contour position with sub-pixel precision, which is more stable than using the local peak position directly.

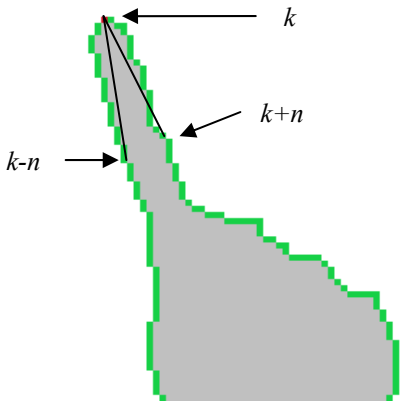


Figure 3.5 – Finding potential peaks along a hand contour. The final fingertip position is computed using a weighted average of neighbouring peaks.

The 2D finger orientation can then be computed by fitting a line through the midpoints along the finger's axis. If k represents the location of a fingertip along the contour, we choose contour point pairs at $k+i$ and $k-i$, for all i between i_{min} and i_{max} (we currently use 8 and 28 for i_{min} and i_{max} respectively). The midpoints of each $k+i$ and $k-i$ contour point can then be computed, which act as support points along the axis of the finger. A line can then be fit through these points to extract the finger's axis as well as the orientation. Figure 3.4d shows the result of fingertip position and orientation detection.

3.2.3.5 Fingertip Labeling

We use a simple heuristic approach to label each of the fingertips that are found in the previous step. The rules are as follows:

- If a single fingertip is detected in the contour, it is always labeled as the index finger under the assumption that this is the finger primarily used for pointing tasks.
- If two fingers are detected, the system assumes they are the thumb and index finger, since these fingers are commonly used in pinching and grasping tasks. To correctly label these two fingers, we use the distance between each detected fingertip along the contour to differentiate between the two. For the right hand, the distance from the index finger to the thumb is larger in the clockwise contour direction than the distance from the thumb to index finger. For the left hand, the opposite is true.
- If three fingers are detected, they are assumed to be the thumb, index, and middle fingers. For the right hand, clockwise distances are computed between each adjacent pair of fingertips in the contour, and the two fingertips with the largest distance are assumed to be the middle finger and thumb. Using this information, the index finger label can be assigned to the fingertip that falls between the thumb and middle finger in the clockwise contour. In a similar manner, the counter-clockwise distances can be used for left hand fingertip labeling.
- If four fingers are detected, they are assumed to be the thumb, index, middle, and ring fingers. Similar to the three finger approach, the largest clockwise distance between neighbouring fingertips in a right hand contour is assumed to be between the ring finger and thumb, which facilitates the labelling of the index finger and middle finger. For the left hand scenario, the counter-clockwise distance can be used instead.

- Finally, if five fingers are detected for a right hand, the largest clockwise distance between neighbouring fingertips is assumed to be between the little finger and thumb. The remaining fingertips can then be labelled by starting at the thumb and then assigning fingertip labels in clockwise order. Counter-clockwise order can be substituted for the left hand scenario.

3.2.3.6 Detecting Contact with the Visual Touchpad

For each camera, the hand detector gives us the *(tx, ty)* position of fingertips in 2D touchpad space, as well as the 2D orientation angle θ of each finger. For fingertips directly on the surface of the touchpad, the positions will be the same regardless of whether we use the pose information from the rectified image from camera 1 or the rectified image from camera 2. However, for fingertips *above* the touchpad surface the positions of corresponding points will be different (Figure 3.6a-c). As shown in Figure 3.7, the disparity d between 2D points can be used to estimate the height z of a fingertip above the touchpad surface, where d=0 represents fingertips directly on the surface and z is proportional to d. Such an approach has the advantage that no knowledge of the intrinsic or extrinsic camera parameters is required. However, for points above the surface, the (tx, ty) values from each camera are inaccurate representations of the actual metric (x, y) position in touchpad space.

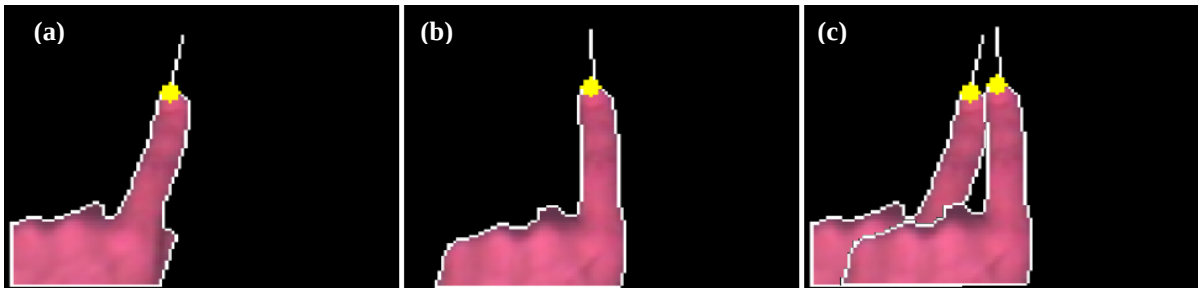


Figure 3.6 – Using disparity for sensing height of raised fingers: (a) Rectified camera 1 view; (b) Rectified camera 2 view; (c) Images overlaid together show corresponding points for raised fingers are not in the same position.

Ideally, what we would like to do is determine the 3D (x, y, z) position of every fingertip in touchpad space so that we can perform manipulations above the surface as well. Since we have a homography from metric touchpad space into image space for each camera, we can perform a coarse calibration of the stereo system using the technique proposed by Zhang

[Zhan99]. This provides us with intrinsic and extrinsic camera parameters which allow us to triangulate 3D fingertip positions both on and above the surface.

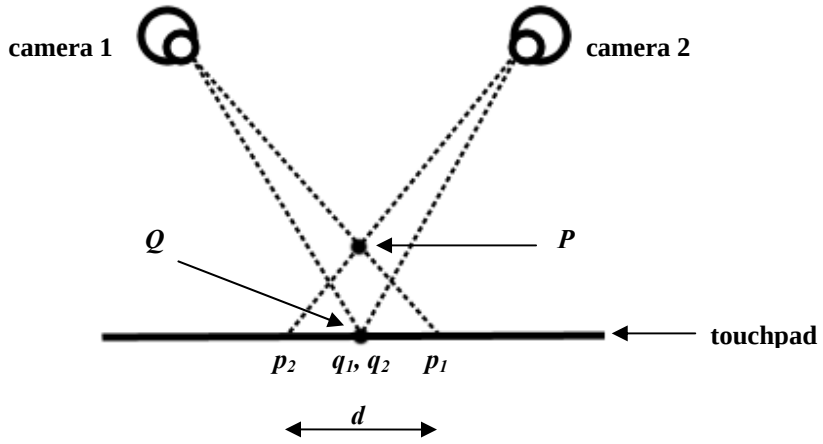


Figure 3.7 – For a point *Q* on the surface of the touchpad, the projected positions q_1 and q_2 on the touchpad will be at the same location in the rectified images. However, for a point *P* above the surface, the disparity *d* between the projected points p_1 and p_2 can be used to estimate the height above the touchpad. Metric 3D positions of points can be recovered by stereo triangulation.

With a full 3D reconstruction, the final output of our hand tracker is a set of (x, y, z, θ) values for each detected fingertip. Therefore, the x and y parameters can be scaled to the dimensions of a screen to simulate an absolute multi-point touch-sensitive surface, while the z parameter can be used to determine (based on a threshold) whether a fingertip is making contact with the surface of the touchpad. Note that we set one of our cameras to be the reference camera, and thus the θ values for each fingertip are extracted from the hand contour associated with that camera's rectified touchpad space image. Additionally, the tracker can also provide temporal information, resulting in a total of five parameters for each fingertip.

3.3 Analysis of System Performance and Limitations

In this section we will provide an analysis of the performance and limitations of the Visual Touchpad system. We first start with a qualitative analysis that examines the performance of the hand and finger labeling system, followed by a quantitative analysis of the 3D position and 2D orientation detection capabilities.

3.3.1 Qualitative Analysis

3.3.1.1 Finger Labeling

Recognition of the index finger positions works well for both the left and right hands with the default threshold values. Figure 3.8a shows an example of the recognition of the pointing gesture when both hands are present in the image. Since the system relies on a minimum hand blob size, however, it is possible that hands which are partially off the edges of the touchpad will not be detected. A possible remedy to this problem is to map a smaller active region (inside of the black rectangle) to the corners of the display instead of using the entire black rectangle. Another limitation of the fingertip detector is the inability to detect fingertip positions when two fingers are contacting one another (since the contour peak detector fails).

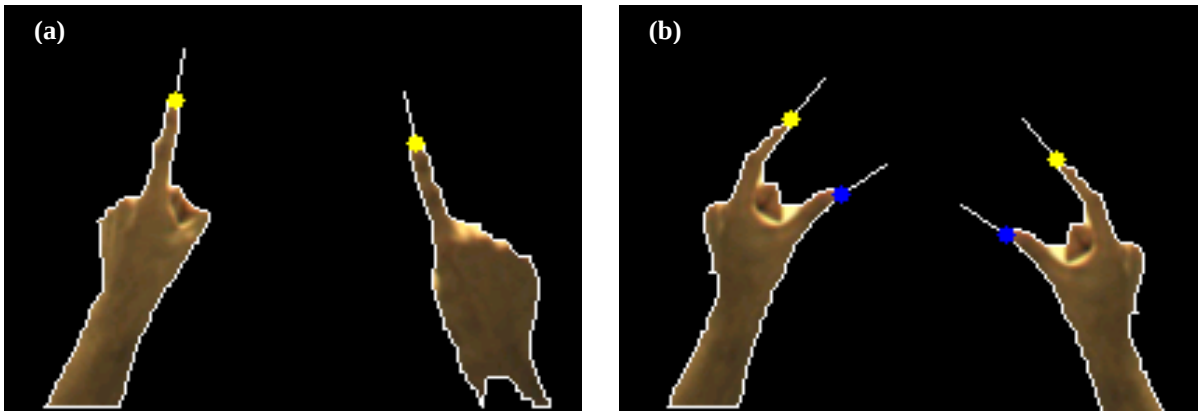


Figure 3.8 – (a) Successful recognition of the pointing gesture for each hand; (b) Successful recognition of the pinching gesture for each hand. A yellow dot is the index finger tip; a blue dot is the thumb tip.

Figure 3.8b shows the successful recognition of the thumb and index finger for both hands. The simple heuristic to disambiguate two or more fingers works well in most instances if we assume that a user is familiar with the posture and gesture set for an application. However, the system is not foolproof. As can be seen in Figure 3.9a, the single outstretched thumb is incorrectly being labeled as the index finger instead. Similarly, Figure 3.9b shows the incorrect labeling of the outstretched index and middle fingers. Based on the simple heuristic for the two fingertip scenario, the index finger is labeled as the thumb, and the middle finger is labeled as the index finger.

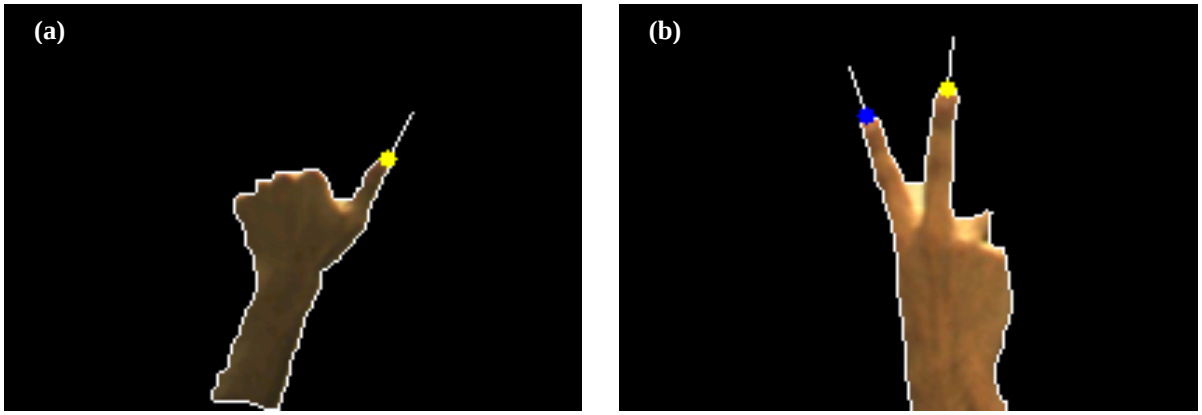


Figure 3.9 – Incorrect labeling of fingertips. (a) If only one finger is outstretched, it is assumed to be the index finger (yellow); (b) If two fingers are outstretched, one is assumed to be the thumb (blue) and the other the index finger (yellow).

3.3.1.2 Hand Labeling

As seen in Figure 3.8, the system can correctly detect the left and right hands of a user, with correct labeling of fingertips based on the label priorities described earlier. However, when a single hand is placed on the touchpad, the system currently assumes it is the right hand. Therefore, as seen in Figure 3.10a, a pinch gesture with a single left hand results in the labeling of the thumb and index finger to be reversed. A related problem occurs if the left and right hands are crossed over one another since the system relies on the centroid of each detected blob to determine the left and right hand labels (Figure 3.10b). A similar misclassification occurs when the palms are shown to the camera, since the system currently assumes that hands will always be seen from the top.

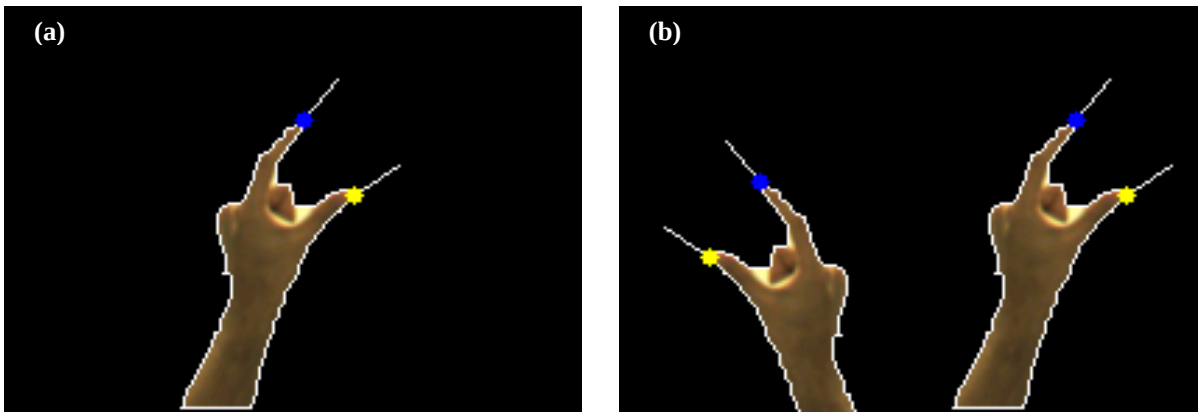


Figure 3.10 – Incorrect labeling of hands. (a) A single hand is assumed to be the right hand, which can lead to incorrect finger labels when a single left hand is seen. (b) If two hands are crossed over one another, the left most hand is assumed to be the left hand, and the right most hand is assumed to be the right hand, which may lead to incorrect finger labeling.

Since the number of hands in the image is determined based on the number of large disconnected blobs, the system cannot correctly label hands or fingers when two hands make contact with one another. As seen in Figure 3.11, the left and right hands are crossed over one another and the blob detector has extracted one large contour that is entirely labeled as the right hand. As a result, the outstretched index finger for the right hand is being labeled as the thumb for a right hand, while the outstretched index finger for the left hand is being labeled as the index finger for a right hand.

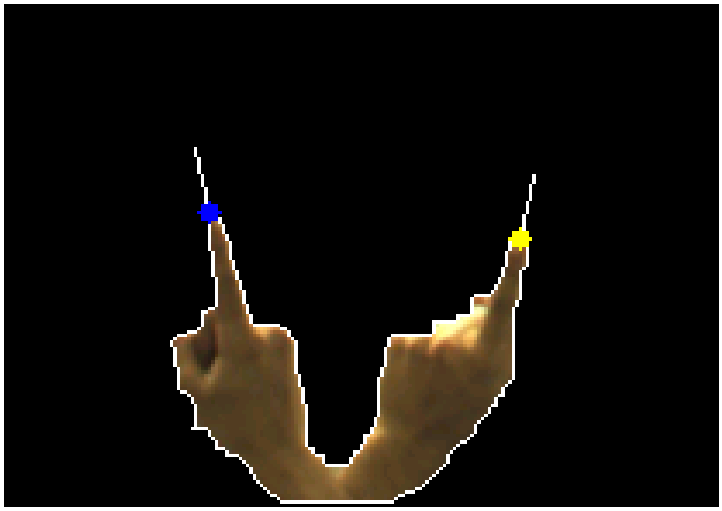


Figure 3.11 – When two hands make contact with one another, the system considers the contour to represent a single right hand.

3.3.2 Quantitative Analysis

In order to assess the accuracy of the position and orientation detection of the Visual Touchpad, we compared the output of our device with the output of a Vicon optical motion tracking system. Three reflective markers were attached to the dorsal side of a single participant's right hand so that they were always visible to the Vicon system as shown in Figure 3.12. Since the reflective markers were light gray in appearance, they did not interfere with the hand segmentation of the Visual Touchpad. The participant was instructed to keep the index finger outstretched, and a pen was taped underneath the length of the index finger to maintain the outstretched posture. The remaining fingers were kept in the palm of the hand to grasp the base of the pen. The reflective markers were positioned so that the axis of the

outstretched index finger could be determined based on the vector between marker 1 and marker 3 (Figure 3.12). The distance between marker 3 and tip of the index finger was measured to be 110mm.

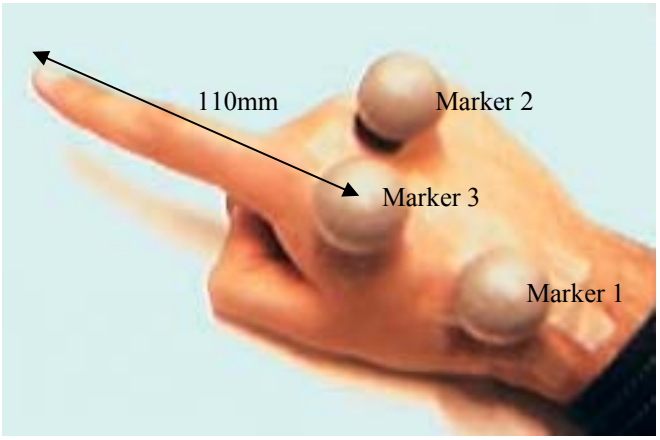


Figure 3.12 – Reflective Vicon markers attached to the hand.

The Visual Touchpad was detected using two standard webcams: a Creative Webcam Pro, and a Creative Webcam Notebook. The cameras were placed 64cm above a 472 x 274mm Visual Touchpad region, with a distance of approximately 30cm between each camera. One camera was positioned so that the black touchpad region was centered in its view and the camera's optical axis was approximately perpendicular to the touchpad. The second camera was placed so that its optical axis was approximately 45 degrees to the touchpad. Both cameras captured the scene at a resolution of 320x240 at approximately 20 Hz.

Since the Vicon system uses its own coordinate space, we manually determined the corners of the black Visual Touchpad region in Vicon space by holding the finger at each touchpad corner and recording the index finger's tip position based on the offset from the markers. Once the Visual Touchpad and Vicon coordinate spaces were calibrated, the participant was instructed to smoothly move the outstretched index finger to random 3D positions over the touchpad region for approximately 3 minutes. The (x, y, z, θ) values from the Visual Touchpad and the corresponding values from the Vicon system were then recorded and output to a data file for analysis. Data capture occurred at the Vicon's refresh rate of 100 Hz, which resulted in approximately 20000 samples. Outliers, which occasionally occurred when

the fingertip reached the edges of the touchpad, were removed before performing the analysis.

3.3.2.1 Position Accuracy

Figure 3.13 plots the X position of the index finger with both the raw Visual Touchpad data and the raw Vicon data. Using the Vicon data as ground truth, the mean error in the X value reported by the Visual Touchpad was -21.83mm, with a standard deviation of 12.97. Overall, the X position computed by the Visual Touchpad appears to be highly accurate, although accuracy degrades slightly as the fingertip approaches the left edge of the touchpad (X < 100mm). Errors increase near the edges of the touchpad primarily due to difficulty in finding the fingertip peaks when the projection of the finger begins to fall outside of the rectified touchpad image for the second camera (due to the camera's orientation with respect to the touchpad).

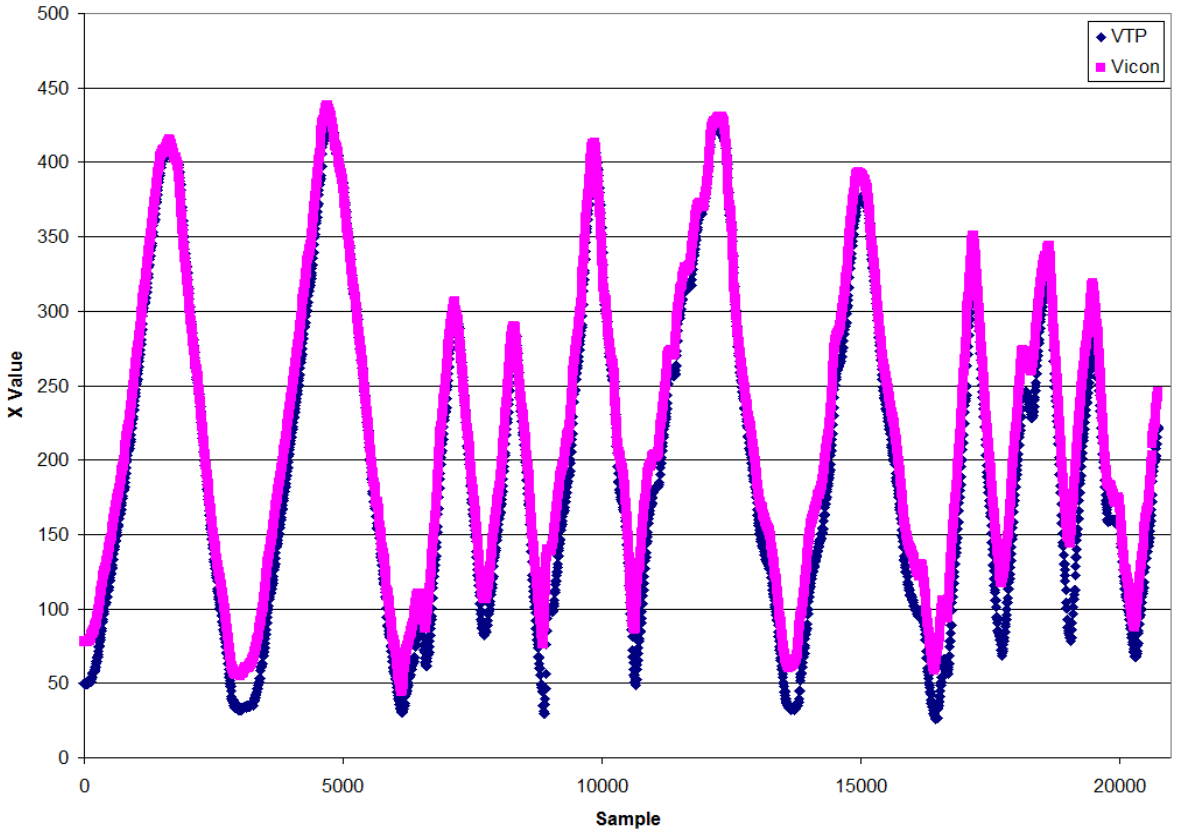


Figure 3.13 – X position accuracy of the Visual Touchpad.

In a similar manner, Figure 3.14 plots the Y position accuracy of the Visual Touchpad. The mean error was 16.72mm with a standard deviation of 12.31, which is comparable to the results along the X axis. Errors were higher as the fingertip approached the bottom of the touchpad since the fingertip detector assumes a minimum contour size for the hand. This suggests that a slightly larger touchpad region that contains a smaller active region inside of it would be desirable.

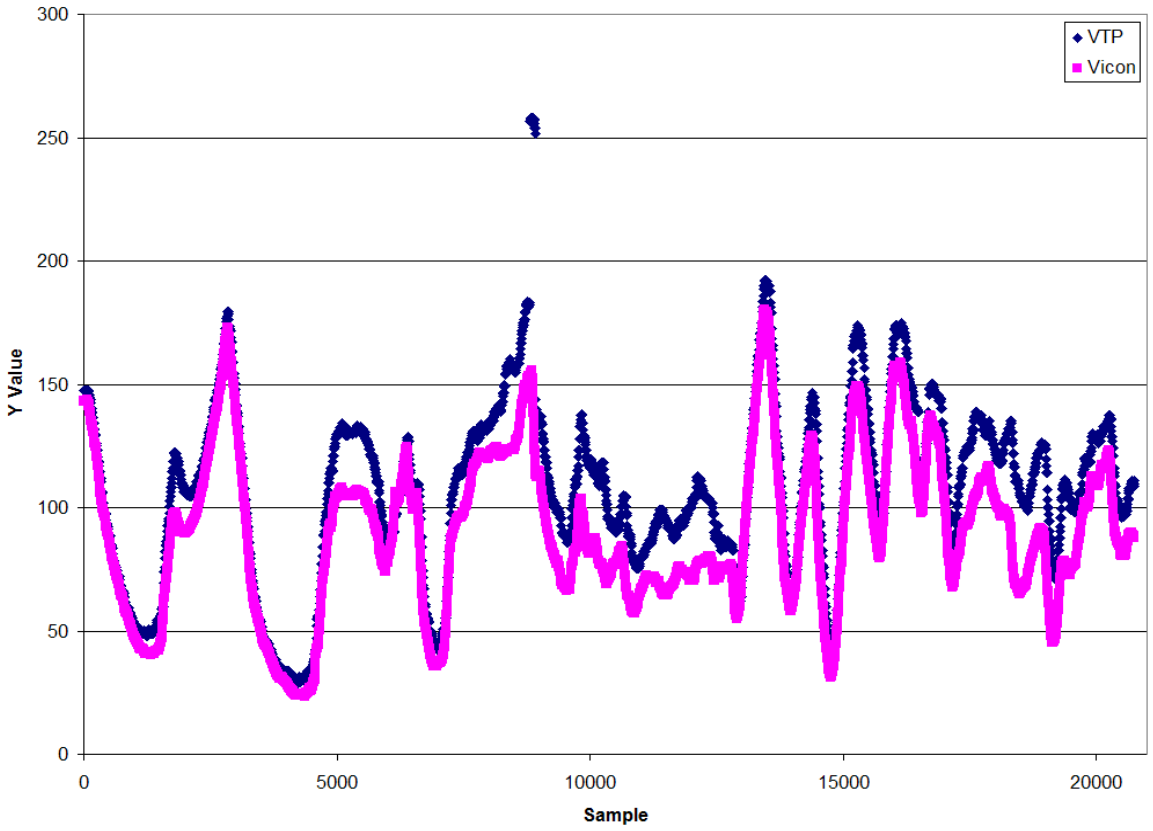


Figure 3.14 – Y position accuracy of the Visual Touchpad.

Finally, Figure 3.15 plots the Z position accuracy of the Visual Touchpad. Mean Z error was -48.54mm with a standard deviation of 24.41, which is significantly higher than both the X and Y results. Nevertheless, based on the plot, it is clear that the Z trajectory reported by the Visual Touchpad well-approximates the Z value reported by the Vicon. However, there appears to be a discrepancy in the scale of the Z values, which suggests that while the direction of the Z axis that is estimated from the coarse calibration of each camera is

sufficiently accurate, its scale is highly imprecise. Figure 3.16 plots the Z values of the Visual Touchpad and the Vicon against one another after removing outliers. The equation of the regression line is y=-5.760 + 0.232*x, with r^2 = 0.946. The RMS error of the Visual Touchpad Z value and the Vicon Z value was 38.10.

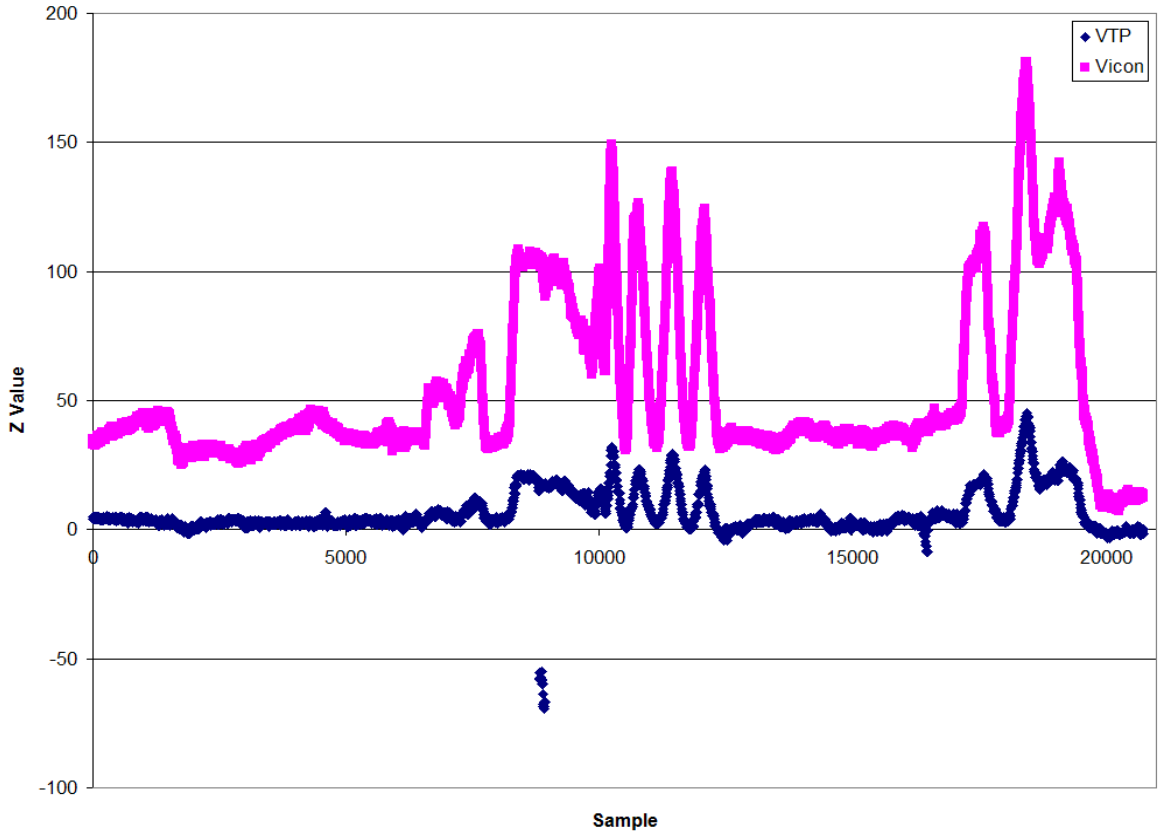


Figure 3.15 – Z position accuracy of the Visual Touchpad.

For the purposes of a touch-sensitive device, however, absolute metric Z information is not critical. What is important is the ability to consistently determine contact information with the touchpad. Based on the Z data in Figure 3.16, a binary touch state for each fingertip can be reliably determined with a threshold of approximately 3cm. Due to the scaling inaccuracy between the Visual Touchpad and actual metric units, this threshold can be set using a simple calibration phase. Alternatively, an intermediate hover or tracking state can be defined at 3cm, with the no-touch state at 6cm. This approach allows the Visual Touchpad to adhere to Buxton's three-state model for touch-sensitive devices which allows for more advanced

manipulations [Buxt90]. As processing power and camera quality continues to improve, we imagine these thresholds will decrease so that detecting two or more intermediate states may also be possible without significantly straining the fingers.

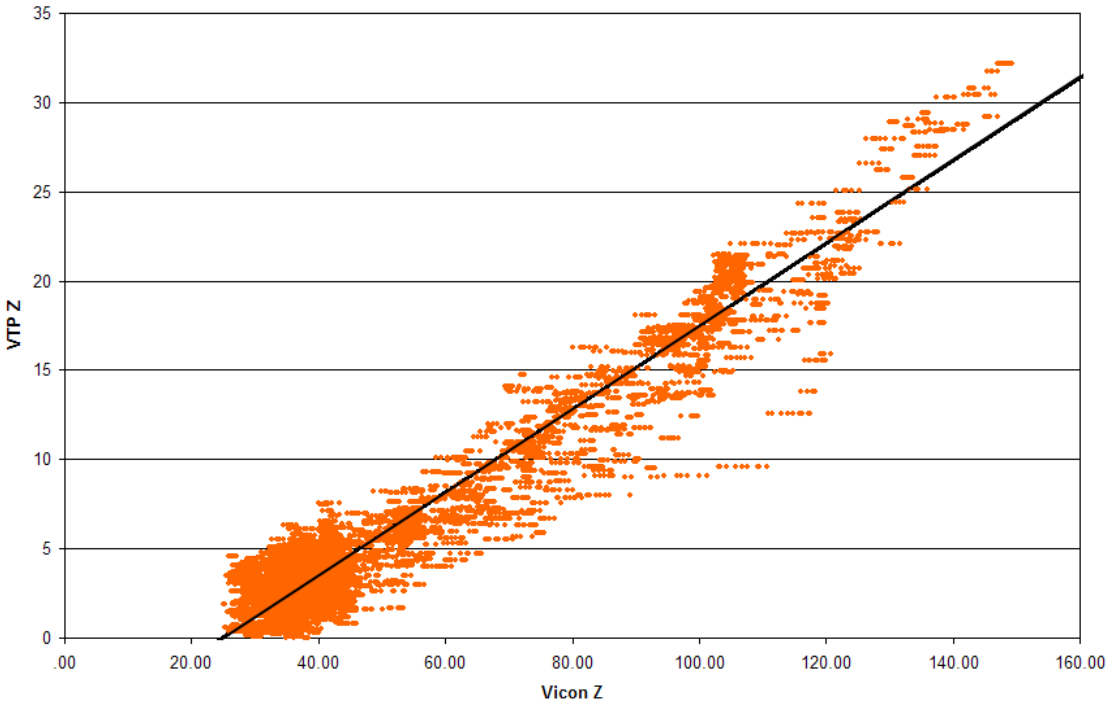


Figure 3.16 – Plot of Vicon Z values versus Visual Touchpad Z values.

3.3.2.2 Orientation Accuracy

We also measured the orientation accuracy of the Visual Touchpad using the Vicon data as ground truth. The axis of the finger was converted into an angle between 0 and 180 degrees, where 0 represented the positive X direction, 90 represented the –Y direction, and 180 represented the –X direction. Figure 3.17 plots the θ accuracy, with mean error of 7.99 degrees and a standard deviation of 5.21. This suggests that the orientation reconstruction is highly accurate. However, based on the plot of the Visual Touchpad data, it is clear that noise is significantly higher compared to the relatively smooth Vicon data.

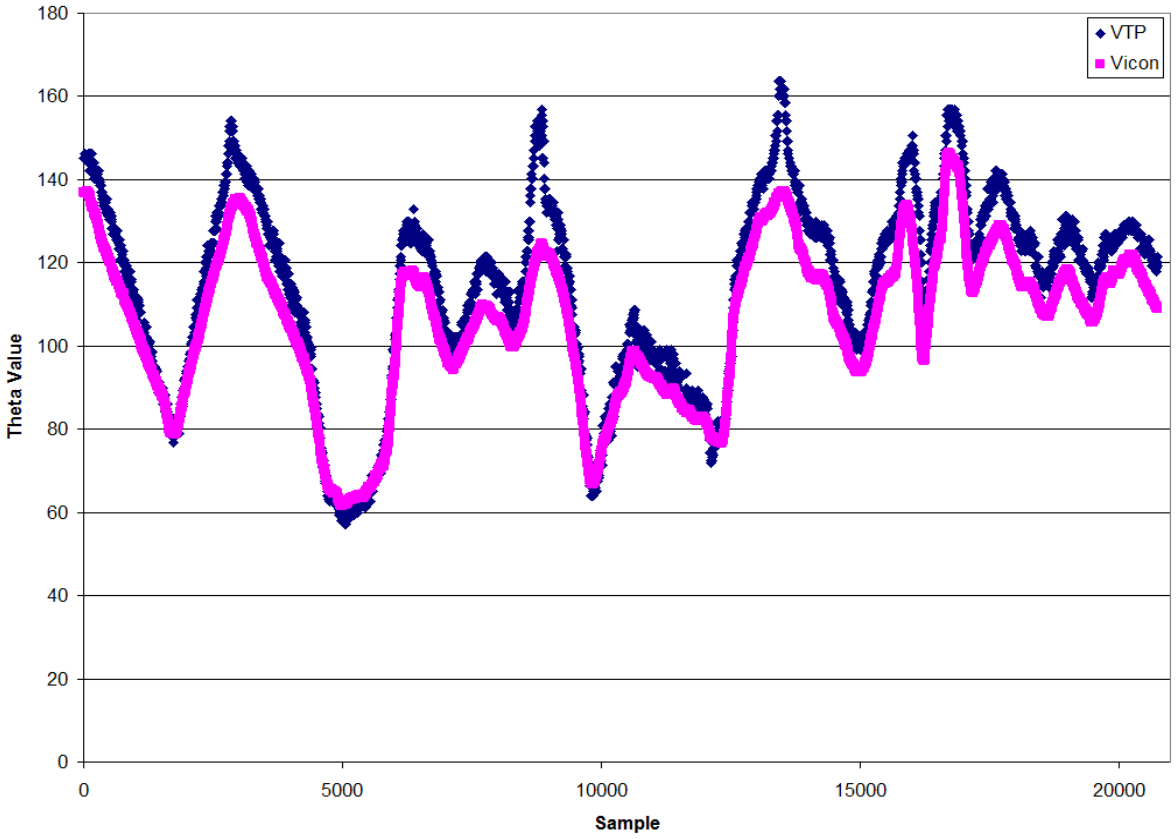


Figure 3.17 – θ finger direction accuracy of the Visual Touchpad.

3.3.3 Other Limitations

While the vision-based aspect of our system provided us with functionality that cannot be performed on standard touch-sensitive hardware (such as finger labels, orientation, and hover), there are still a few downfalls. In particular, the arch-enemy of vision algorithms is darkness, which prevents the system from being used as is in low-light environments. Ultimately we imagine that one could build a stand-alone vision-based device that consists of a glass touch surface, with an array of infrared cameras and lights embedded underneath the glass. This would not only resolve the issue of dark rooms, but it would also remove the need to have cameras hanging overhead or mounted above the display. A related problem with most vision algorithms is the difficulty when segmenting objects under varying lighting conditions and shadows. Since we use simple background subtraction to extract the hands from above a black surface, our system is quite reliable under a wide range of illumination conditions. Another possibility is to combine the strengths of our system with a touch-

sensitive surface such as the DiamondTouch or SmartSkin for more robustness. This would increase the positional accuracy of the system while still providing benefits of the vision system such as multi-layer gesture recognition and accurate finger orientations.

Although our current embodiment assumes that the input and output spaces are separate, it would be interesting to investigate how the Visual Touchpad technology could be adapted to support direct-touch devices. For example, with a front or rear-projected direct-touch display, a more advanced background model would be required since simple black background subtraction would not be sufficient to accurately segment out the hands or fingers. Similarly, with a front-projected display, the system would additionally have to handle projections onto the surface of the hand as well as shadows cast onto the touch-surface. Cham et al. [Cham03] investigated how a camera combined with a front-projected multi-projector display could be used to eliminate shadows cast by occluding objects as well as suppress light on such objects during slide presentations, so this could be a good starting point to address some of the limitations that currently prevent the Visual Touchpad from being used in a direct-touch scenario. Other possibilities include using infrared cameras to help segment hand pixels from the background, or using depth-sensing cameras to determine the 3D position and shape of fingers over the surface.

Vision algorithms are highly susceptible to image noise, particularly when using the low-cost sensors that can be found in many webcams. The Visual Touchpad is sensitive to image noise during contour detection, since detected edges may fluctuate by a few pixels from frame to frame. As a result, the detected fingertip positions and orientations are susceptible to a small amount of jitter, which can be seen in the X, Y, Z, and θ plots described earlier. Position filtering techniques can help reduce the amount of jitter, but at the expense of increased lag.

Vision-based tracking algorithms that track local features from frame-to-frame are prone to tracking failure when features become lost. Since the Visual Touchpad does not use any temporal information to extract fingertip positions, fast hand movements do not cause the tracker to fail. However, depending on the quality of the cameras, fast movement may cause motion blur which does confuse the fingertip detector. In such situations, the system simply

uses the last valid fingertip positions and attempts to recover the actual positions in subsequent frames (before some predefined timeout expires). Additionally, PCs do not currently provide any mechanism to synchronize the capture of frames from multiple cameras. As a result, if the capture rate of one camera is significantly lower than the other (which often happens when different camera brands or models are used) the triangulated fingertip positions will jitter considerably.

3.4 Summary

In this chapter we presented the Visual Touchpad, a prototype vision-based input device that allows for extracting rich information such as finger labels, hover, and orientation, each of which are difficult to determine on existing multi-finger input devices. While the quantitative and qualitative analyses demonstrate a number of limitations of the device, the performance is sufficient for using it as an easy-to-use, low-cost platform for rapidly building prototype multi-finger user interfaces. We perform such design explorations in the following chapter.

Chapter 4

Design Explorations of Multi-finger Input

4.1 Introduction

In this chapter we design three applications which explore how multi-finger input can be used in real-world scenarios. The designs also serve to demonstrate how the low-cost Visual Touchpad described in Chapter 3 can be applied to practical problems without relying on more expensive and intrusive optical or glove-based tracking systems.

4.2 Fluid Picture Manipulation

Our first design exploration implements a simple picture manipulation application that uses the Visual Touchpad as a multi-point input device to merge the functionality of the keyboard and mouse in a standard desktop scenario. A number of images are scattered around a virtual canvas, and using hand gestures the user is able to move/rotate/scale the images, query object properties, pan/rotate/zoom the view, draw onto the canvas, stretch images, and annotate images with text. Using a simple set of multi-finger postures and gestures, we show that the Visual Touchpad can be used to perform a variety of common GUI operations in a fluid manner.

4.2.1 System Overview

For the picture manipulation application, we use the same Visual Touchpad setup from Chapter 3 where the active touch area measures 472 x 274mm. The touchpad coordinates are mapped to the entire display in an absolute manner, and we use depth information to denote two states for each fingertip: contact and no contact. Contact is defined as 0 to 3 cm, and no contact is defined as above 3cm. The overall system runs on a 2GHz Pentium4 computer at approximately 20Hz using two web cameras capturing at a resolution of 320x240 pixels each.

4.2.1.1 Postures and Gestures

Given the output of the Visual Touchpad's hand tracker, it is extremely simple to detect the four static postures depicted in Figure 4.1. The pointing posture is simply the index finger held straight out in some direction. The pinching posture involves setting the thumb and index finger as if something is being held between them, with the thumb and index finger pointing in relatively the same direction. The L-posture is a variation of the pinching posture, where the thumb and index finger are pointing in approximately orthogonal directions. For both the pinch posture and L-posture we can overload the recognition system with variations such as both fingers touching the touchpad surface, both fingers not touching the surface, or one finger on the surface and one finger off the surface. Finally the five-finger posture is simply holding out all fingers so that the hand detector can clearly identify all fingertips.

Along with the static postures, our system can also detect gestures using temporal information. To demonstrate this capability, we currently detect a holding gesture (for all postures), a double-tap gesture (for the pointing posture), and an X shape gesture (also for the pointing posture).

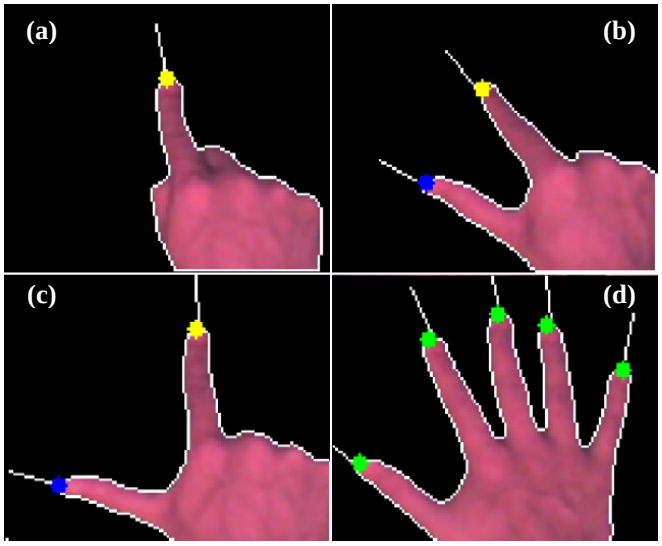


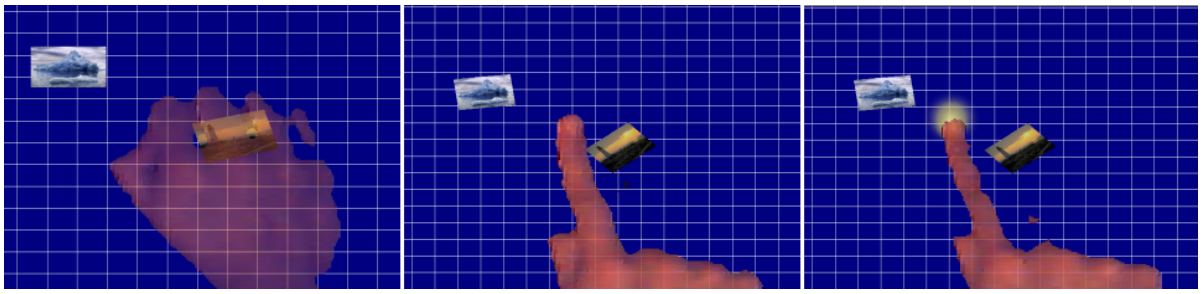
Figure 4.1 – Posture set: (a) Pointing; (b) Pinching; (c) L-posture; (d) Five-finger posture.

4.2.1.2 Hand Augmentation

Various researchers have recognized the value in augmenting displays with overlaid live video proxies of the body for compelling visual feedback. For example, the Videowhiteboard [Tang91] and the ClearBoard [Ishi92] both displayed overlaid video of a collaborator working on a shared planar workspace. Buxton [Buxt92] presented a good discussion of these and related earlier systems, with a primary focus on shared workspaces and collaboration. A more recent system called the VideoPointer [Rous01] proposed the overlay of a user's hand as an expressive remote pointing device. Each of these works use overlaid live video primarily for remote awareness in applications such as teleconferencing, and do not make use of the video proxy as an active user input.

The Video FaceTop proposed by Stotts et al. [Stot03], however, overlays the desktop with a live video reflection of the user, which can be used to manipulate onscreen widgets. Building on this idea, we propose augmenting the user's hands directly into the graphical interface, using the live video of the segmented hand regions from the reference camera as a visual proxy for direct manipulations. The advantage of this approach is that a user feels more connected to the interface in a manner similar to tabletops or touch-screens, but while using an external display such as a monitor. The other advantage is that by rendering the hands as images, we can apply other special effects such as transparency to help overcome the

occlusion problem, as well as attach visual annotations onto the hand such as mode or state information. Figure 4.2 shows an example of a hand being augmented onto a graphical interface.



(a) **(b)** **(c)**

Figure 4.2 – Hand augmentation: (a) No fingers; (b) Finger above surface; (c) Finger contacting touchpad

Note that the size of the hand on the screen is dependent upon the size of the touchpad, since we stretch the entire hand image from the reference camera's rectified touchpad image onto the whole screen. Therefore, the larger the touchpad, the smaller the hand appears on the screen, and vice-versa. Thus the dimensions of the touchpad should be proportional to the size of the display.

When no fingers are detected by the hand tracker, any hand blobs are rendered with 50% opacity (Figure 4.2a). As soon as any fingers are detected, each fingertip is drawn at 85% opacity with gradual falloff to 50% opacity using a fixed falloff radius (Figure 4.2b). This allows the hand to come into focus when the user is performing some action. Additionally, when a fingertip is determined to be in contact with the touchpad, a yellow highlight is rendered beneath it for visual touch feedback (Figure 4.2c).

One limitation of this augmentation is the potential offset between the 3D position of a fingertip and the corresponding visualization of the hand on the screen. As fingertips are raised above the surface, the (tx, ty) position of a fingertip in the reference camera's rectified touchpad image will not necessarily be at the same position as the (x, y) fingertip coordinates from the reconstructed 3D position. While the offset is not significant for interactions that

occur only a few centimetres above the surface, a possible remedy is to use the (tx, ty, z, θ) parameters instead of the reconstructed (x, y) values.

4.2.2 One-handed Techniques

4.2.2.1 Object Selection/Translating/Rotating/Query

To select an image on the canvas, a single hand in a pointing posture can be positioned so that the fingertip is within the bounds of the object, with the fingertip touching the surface of the Visual Touchpad. When the finger makes contact with the touchpad a yellow glow appears around the fingertip. Additionally, the borders of the selected image become green to signify that it has been selected. To deselect the object, the user simply raises the fingertip up from the touchpad surface until the yellow glow disappears.

Once an object has been selected, it can be simultaneously translated and rotated. Translation is controlled by simply moving the finger in the appropriate direction. The image then remains attached to the fingertip. Similarly, the orientation of the finger controls the rotation of the object, with the centre of rotation being the fingertip position. Figure 4.3 shows an image being translated and rotated using the pointing gesture.

To query an object for information (such as file name, image dimensions, etc) we use an approach similar to tooltips found in graphical interfaces such as Windows. By simply holding a pointing posture for one second inside the boundaries of an image, but without touching the touchpad surface, a small query box is activated. Moving the finger out of the image dismisses the query box.

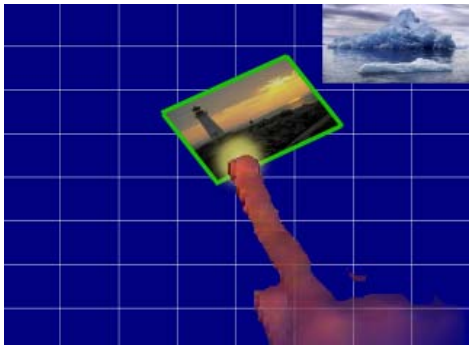


Figure 4.3 – Image translation and rotation with a single finger.

4.2.2.2 Group Selection/Copy/Paste/Delete

To select a group of images for operations such as copying or deleting we can make use of the double-tap gesture. By double-tapping on an image a yellow highlight appears around it signifying that it has been added to the current group. To remove a selected image from the group we simply double-tap it again. A single tap in any empty canvas location causes the entire group of objects to be deselected.

The selected group is always the set of objects in the clipboard so there is no need to explicitly perform a *copy* operation. To paste the selected group of images we use the L-posture with both fingers above the touchpad surface. The index finger position defines the centre of the selected group, and translation or rotation of the L-posture can be used to place the group in the desired location. To finalize the positioning the user simply places both the index finger and thumb onto the touchpad surface. After the paste operation, the new set of images becomes the active group selection. Note that the second hand can be used to navigate the canvas viewpoint simultaneously (as described in the next section). To *cancel* the paste operation the user can touch the thumb and index finger together without touching the touchpad surface. To *delete* a selected group of images a user draws an X in an empty part of the canvas.

4.2.2.3 Canvas Panning/Rotating/Zooming

To control the canvas viewpoint we use an approach similar to the SmartSkin map viewer [Reki02]. Using a pinching posture, where the thumb and index finger are in contact with the surface of the touchpad, the user can simultaneously control the position, orientation, and zoom level of the window into the canvas. The idea is that as soon as two fingers make contact with the touchpad, they become “attached” to the corresponding positions within the canvas. Moving the hand around the canvas while maintaining the pinch posture causes the window into the canvas to move in a similar direction. To rotate the entire canvas, the hand can be rotated while the pinch posture is maintained. The centre of rotation is thus defined as the midpoint between the tips of the thumb and index finger. Finally, bringing the fingers closer together while still touching the surface causes the view to be zoomed out, while moving the fingers further apart causes the view to be zoomed in. The centre of zoom is defined as the midpoint between the thumb and index finger. In all cases, when translation,

rotation or zooming becomes difficult due to the hand ending up in an awkward pose, the operation can be continued by simply raising the fingers off the touchpad surface, adjusting the hand to a comfortable position again, and then continuing the viewpoint control (this is commonly referred to as *clutching*). Figure 4.4 shows an example of a pinch posture controlling the zoom level.

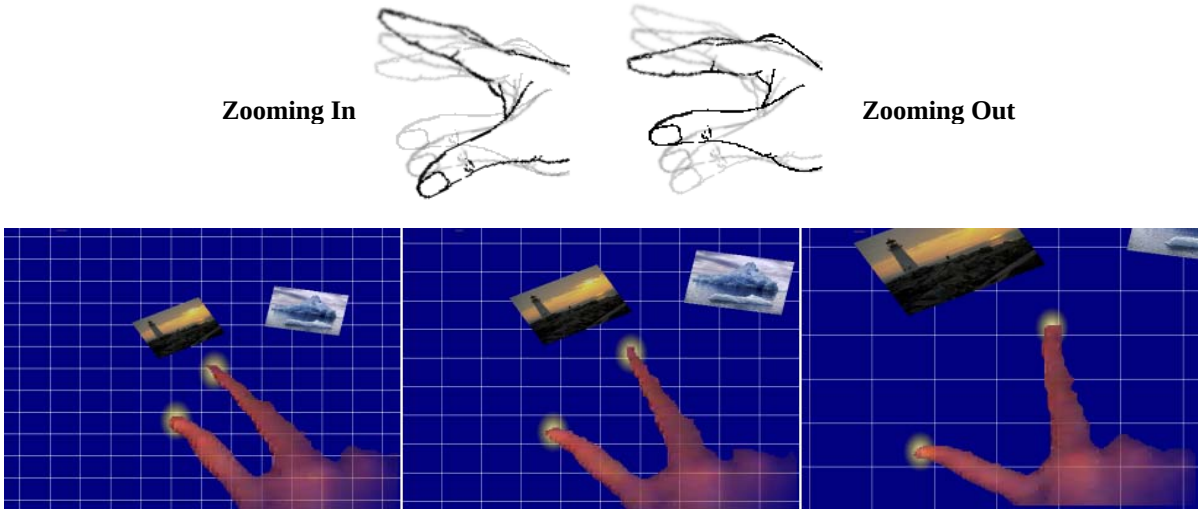


Figure 4.4 – Canvas zoom control using two fingers.

4.2.2.4 Navigation Widget

While the canvas viewpoint control described above works well for small adjustments of the canvas, it is inefficient when large-scale viewpoint changes are required. Since we are able to recognize postures and gestures above the touchpad surface, we propose a navigation widget that can be used for continuous scrolling of the viewpoint. To activate the widget the user holds a pinch posture steady for one whole second above the surface of the touchpad. Once activated, the system captures the midpoint between the thumb and index finger as the "centre" position. A navigation arrow then appears between the thumb and index finger, with a line connecting the current midpoint between the thumb and index finger to the "centre" position (Figure 4.5).

The widget then acts much like a joystick, where translation in any direction away from the "centre" causes the viewpoint to translate in that direction, with scrolling speed dependent upon the distance of the widget from the "centre". Canvas zooming can also be performed, by treating the navigation widget as a dial, where the "zero" rotation is the finger orientation

at the "centre" pose. Therefore, rotation of the fingers in a clockwise direction causes the view to be zoomed in, while a counter-clockwise rotation causes the view to be zoomed out. The amount of rotation from the "zero" defines the speed of the zoom. To deactivate the widget, the user can simply pinch the fingers together completely.

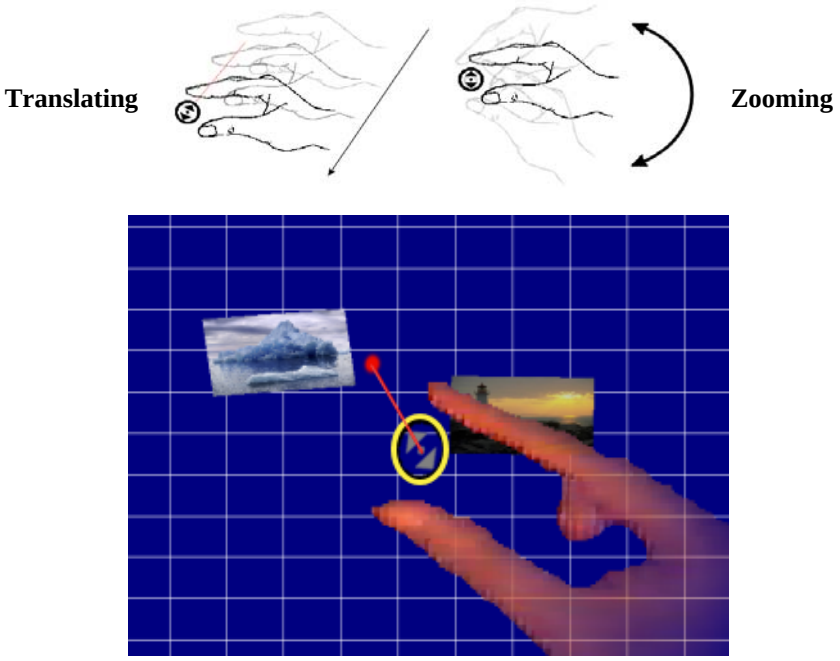


Figure 4.5 – Navigation widget for scrolling and zooming.

4.2.3 Two-handed Techniques

4.2.3.1 Pie Menu

Asymmetric-dependent tasks, as proposed by Guiard [Guia87], are those in which the dominant (D) hand moves within a frame of reference that has been set by the non-dominant (ND) hand. Therefore, the ND hand will engage in coarse and less frequent actions, while the D hand will be used for faster, more frequent actions that require more precision. Kabbash [Kabb94] showed that such asymmetric-dependent interaction techniques, where the action of the D hand depends on that of the ND hand, give rise to the best performance since they most closely resemble the bimanual tasks that we perform in everyday life.

We follow such an asymmetric-dependent approach for our pie menu system that is used to select various options. To activate the pie menu the user performs a double-tap gesture using

the ND hand. The pie menu (with a small hollow centre) is then displayed, centered at the ND hand's index finger. If the user maintains contact with the touchpad surface, the pie menu will remain centered at the index finger. If the index finger is raised from the surface, the pie menu will remain at the previous position, thereby allowing the user to select menu options with a single-tap. Another double-tap in the hollow centre is used to deactivate the pie menu. To illustrate the functionality of the pie menu, we implemented a simple drawing tool that allows the user to "finger-paint" onto the canvas. The pie menu consists of the following options: *drawing mode*, *draw color*, *draw size*, *draw shape*.

The *drawing mode* option acts as a toggle switch. When selected, the D hand's fingertip becomes a paintbrush, with an appropriate cursor drawn at its tip. The user can then paint strokes with the D finger when it is in contact with the touchpad surface.

By selecting the *draw color* option, a color palette is presented to the user. Moving the ND fingertip within the palette (while making contact with the touchpad surface) sets the color of the D hand's fingertip. To deactivate the color palette the user simply moves the ND fingertip out of the palette area.

The *draw size* menu option allows the size of the paintbrush tip to be modified. A slider appears when the option is selected, which can be modified by "dragging" the slider handle using the ND finger much like many 2D GUIs. The slider is deactivated by moving the ND finger outside of the slider's rectangular border.

Finally, the *draw shape* menu option allows the user to change the shape of the paintbrush tip. Four simple shapes are currently implemented as shown in Figure 4.6. Unlike traditional painting tools, ours allows for simultaneous control of not only the brush tip's position but also the tip orientation, thereby allowing for interesting calligraphic effects.

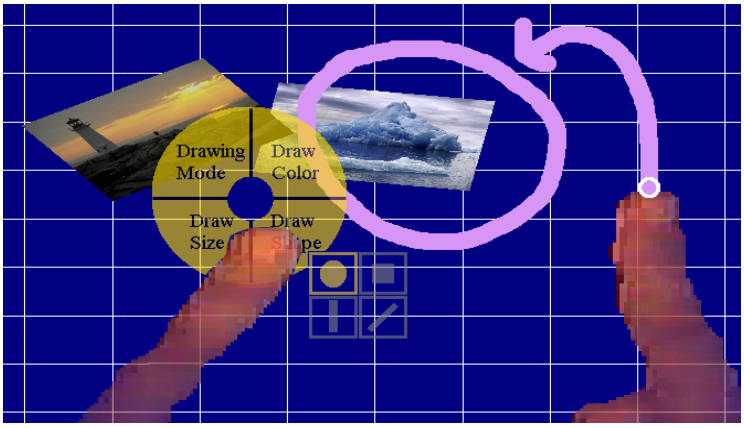


Figure 4.6 – Pie menu for finger painting.

4.2.3.2 Image Stretchies

Kurtenbach et al. [Kurt97] introduced an interaction technique called "two handed stretchies" that allow primitive shapes to be simultaneously translated, rotated and scaled using two rotation-sensitive pucks on a tablet surface. The Visual Touchpad is also capable of such techniques using two-handed postures instead of pucks.

One hand with a pointing posture selects an object as usual. The position of the fingertip is then "locked" onto the selected image. The second hand then selects another position within the same image, and that position becomes "locked". Translating both fingers at the same rate and in the same direction allows for the image to be moved. However, translating the fingers in different directions or at different speeds will cause rotation and scale changes. The idea is that the two "locked" finger positions will always represent the same pixel in the image. While we currently do not make use of finger orientation during this stretch operation, one possibility is to use it as a simultaneous twisting parameter to allow the image to be warped.

4.2.3.3 Virtual Keyboard

Many applications such as presentation tools, drawing tools, or web browsers require frequent switching between text entry (keyboard) and navigation (mouse). Virtual keyboards [Kols02] are one approach to making text entry and navigation more fluid. By rendering a graphical layout of a keyboard on the screen, a user does not have to switch between input devices and can instead focus more on the desired task. Additionally, virtual keyboards can be reconfigured to different layouts based on a user's personal preferences. The downfall

with most virtual keyboards is that they rely on single mouse clicks to simulate key presses, resulting in slow text entry.

Motivated by the familiarity and reconfigurability of virtual keyboards, we have implemented an onscreen *QWERTY* keyboard for the Visual Touchpad that can be used to make textual annotations on our image canvas. To activate the virtual keyboard, a user makes a five-finger gesture with both hands over the touchpad (Figure 4.7). This gesture simulates putting the hands onto "home-row" on a real keyboard. The virtual keyboard is then rendered transparently on the screen, with the hands rendered over top. By default, the text entry cursor is placed at the canvas location corresponding to the middle of the screen, above the virtual keyboard. Letters can then be entered by simply touching the appropriate virtual keys with the fingertips. The virtual keyboard is deactivated by pressing the virtual "Escape" key. Note that the mapping between the touchpad and the virtual keyboard is not dependent on the canvas window settings. Instead, the size and position of the virtual keyboard is fixed to some predetermined absolute region of the touchpad and a corresponding region of the screen so that the spatial layout of the keys remains constant.

By rendering the hands and keyboard together on the display, users do not have to divert their visual attention away from the onscreen task. Additionally, by using the same input surface for both text-entry and GUI navigation, the experience is much more fluid compared to traditional keyboard-mouse configurations. It is worth mentioning, however, that the current implementation does not allow for extremely fast text entry, largely due to the limited speed of our camera capture and image processing operations.

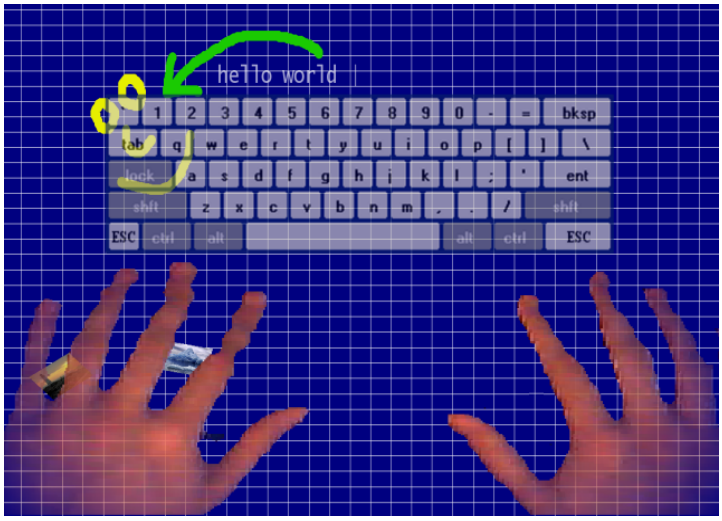


Figure 4.7 – On-screen multi-finger virtual keyboard.

4.2.4 Discussion

Our picture manipulation application demonstrated how the Visual Touchpad could be used for fluid two-handed and multi-finger gestural interactions much like those available on more expensive tabletop displays or touch-screens. By augmenting the live images of a user's actual hands directly into the graphical interface, our Visual Touchpad begins to provide a more compelling "hands-on" experience similar to tabletops or touch-screens while the use of transparency during augmentation avoids the occlusion problems associated with these other devices.

As an initial assessment of the usability of the system, we gathered informal user feedback from six graduate students in our research lab. Each user was given a brief introduction to the posture and gesture set and the various manipulation operations, followed by 10 to 15 minutes of supervised free-form exploration.

All users found the posture and gesture based manipulations to be easy to use, with descriptions such as "cool", "neat", and "fun" to describe the overall system. One of the first things many people were impressed with was the ability to see their own hands on the screen, and as a result they found the direct manipulation techniques to be very compelling.

The asymmetric two-handed pie menu required a quick introduction in most cases, but afterwards all users found the pie menu to be easy to use. Although our pie menu only has four options on it, we tried a quick experiment to see if hand transparency made any difference when portions of the menu ended up beneath the hand. Three users were given a version of the software with a fully opaque hand, while the other three were given a version with a transparent hand. As expected, it was observed that the opaque hand users would frequently move their hand off to the side of the pie menu if an option's title was occluded by the hand, while we did not see this with the transparent hand users. A more extensive study is required to accurately determine how effective our transparent hands are against occlusion, but these preliminary observations are encouraging.

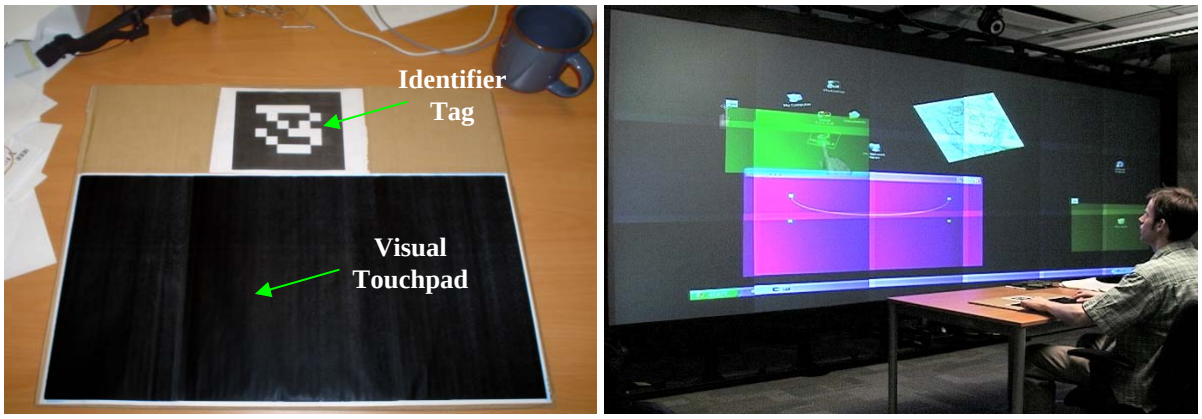
While many users liked the idea of fluidly switching between navigation and text-entry modes, most felt that the virtual keyboard had some drawbacks. Most notably, it was felt that the lack of tactile feedback during keypresses made text entry awkward and prone to errors, since it was difficult to determine key boundaries. One user suggested using some more cues to signify which key was about to be pressed, such as highlighting the most likely key based on the trajectory of the fingertip, or generating audible key clicking sounds. Another complaint with the virtual keyboard was that it occupied a significant amount of screen real estate. An interesting suggestion was to gradually increase or decrease the transparency of the virtual keyboard based on a user's typing speed or idle time, under the assumption that a fast typist has memorized the spatial arrangement of the keys and does not need to see the keyboard as much.

4.3 Large Display Interactions from Afar

The increased screen real estate provided by large wall displays allows for sophisticated single- and multi-user applications that cannot be easily accommodated with standard desktop monitors. However with this larger work area come a number of challenges, particularly from a user interface perspective. While many innovative techniques have been proposed in the literature to deal with the difficulties in quickly accessing all parts of a large display, the majority focus on within arms-reach interactions that assume users will be standing close to the screen [Guim01, Khan04, Myna99, Pede93]. However, consider a

single-user design task that requires the visualization capabilities of a large display but also demands long hours. Similarly, consider a collaborative discussion where users gather around a large conference room table but also frequently need to display things on a large screen for others to see. In these *distant-contiguous* large screen situations [Swam97], allowing the users to interact from the comfort of their chairs seems desirable. While a few such from-afar techniques have been proposed in the literature [Izad03, Joha02a, Joha02b], many still assume mouse-based input and thus fast navigation and target acquisition tasks are still relatively inefficient compared to many arms-reach techniques.

In this section we develop several one- and two-handed multi-finger interaction techniques that support efficient large wall interactions from a distance, whereby a user is seated comfortably in front of the display at a desk or conference room table. Using the Visual Touchpad we explore techniques that allow for a direct manipulation experience on large wall displays using finger manipulations and gestures, similar to a table-top display or touch-screen. We build upon the Visual Touchpad by attaching a small identification tag above the black rectangular region, while two cameras mounted over-head are used to capture live video of the hands and black regions for the real-time vision processing. Figure 4.8a shows our modified touchpad. By attaching unique tags to different touchpads, the system also allows multiple users (each with their own touchpad) to be easily detected. Figure 4.8b shows the actual touchpad in use with a large projection display.



(a) **(b)**

Figure 4.8 – (a) The touch sensitive surface consisting of a unique tag and a solid black colored touch region; (b) A user working with the system on a large rear-projection display.

4.3.1 Issues in Large Display Interaction

Reaching distant targets and navigation of the entire display space are two of the major issues involved in interaction with large (> 10') upright wall displays. As such, there is a large body of literature that investigates these difficulties and proposes some effective solutions [Guim01, Myna99, Pede93]. For example, Bezerianos and Balakrishnan presented a tool called 'Vacuum' for quick access to distant items [Beze05]. The user controls the area of influence of the tool so that distant objects that fall within the area of influence are transported closer to the user for easy selection. Similarly, Khan et al. introduced a widget called 'Frisbee' that uses the concept of a telescope to create a portal to another part of a large display for accessing remote objects [Khan04]. Other techniques such as Drag-and-Pop and Drag-and-Pick [Baud03] can be used for quickly activating distant icons on a graphical desktop, while shuffling and throwing [Geib98] or flicking [Wu03] allow objects to be moved to an approximate location at a specified distance or at the edge of the display.

The majority of these interaction techniques are suited to up-close, pen-based interactions in order to minimize a user's physical movements while standing in front of a large wall display. A number of researchers have also addressed the navigation and target acquisition issues when interacting from a distance. In the Pointright [Joha02b] and i-Room [Joha02a] systems the user can use a standard mouse as the input device and move the cursor across the entire display (consisting of different screens) seamlessly as though they were a single surface. Since they mainly focus on the problem of device-display integration, fast display navigation has not been addressed in detail. Khan et al. presented a technique called 'Spotlight' which allowed a user to control a large highlighted region across a large display from afar in order to direct the visual attention of an audience during a presentation [Khan05]. While this technique has been found to be better than a regular cursor for highlighting targets, it is not clear how it should be used for reaching them efficiently.

Various vision-based techniques have been used for interaction with large scale displays. For example, the systems presented in [Davi02] and [Kirs98] track a laser pointer and use it as an input device which facilitates interactions from a distance. While the laser pointer provides a very intuitive way to randomly access any portion of the wall sized display, natural hand-

jitter makes it difficult to use for precise target acquisition tasks, particularly for smaller targets. Moreover, ordinary laser pointers have only two degrees of freedom which limit their use for complicated tasks. The VisionWand system [Cao03] uses simple computer vision algorithms to track the colored tips of a simple plastic wand to interact with large wall displays both close-up and from a distance. A variety of postures and gestures are recognized in order to perform an array of interactions. A number of other systems use vision to track bare, unmarked hands using one or more cameras, with simple hand gestures for arms-reach interactions. For example, the Bare-Hand system [Hard01] uses hand tracking technology to transform any ordinary display into a touch-sensitive surface. Similarly, the Touchlight system [Wils04] uses two cameras to detect hand gestures over a semi-transparent upright surface for applications such as face-to-face video conferencing or augmented reality. The major advantage of such vision-based techniques is their ability to track multiple fingers uniquely, which allows for more degrees of freedom when compared to standard input devices such as a mouse. However, this advantage of vision-based techniques has not yet been fully leveraged for interactions with wall-sized displays.

While horizontal touch-sensitive surfaces such as the DiamondTouch [Diet01] and SmartSkin [Reki02] could be used to interact with large upright wall displays from afar in a manner similar to the touch surfaces found beneath many laptop keyboards, there are some shortcomings. The SmartSkin, for example, requires the hand to be in relatively close contact to the surface in order for a complete 2D hand image to be detected. As a result, it is difficult for an application to disambiguate which fingers are making contact with the surface. Therefore, when attempting to use such touch-sensitive surfaces for large wall interactions, the finger ambiguity and lack of 2D hand information makes it difficult for a user to visualize how the hand is being mapped to display space.

In short, most of the present interaction techniques for wall-sized displays are limited to up-close interactions using a pen or direct touch, while the limited number of systems allowing interaction from a distance suffer from one or more of the following issues: limited degrees of freedom, lack of visualization of degrees of freedom, inability to differentiate between the two hands and between fingers, or lack of proper balance between quick navigation and

precise target acquisition. Based on these shortcomings, we have designed a vision-based bimanual interaction system that allows for quick navigation and precise target acquisition on large wall displays from afar using multi-finger manipulations and gestures.

4.3.2 Design Principles

In designing fluid interactions for a large wall display for users seated at a table, we have considered the following design issues:

Leverage both hands for multiple degrees of freedom: One of the benefits of large touch screens or tabletop displays is the natural direct manipulation experience they provide, as well as their potential for more complicated interactions using multiple fingers. We leverage this aspect of touch-screens and tabletops by using the Visual Touchpad as our base input device, since it allows two-handed multi-point input as well as the ability to transparently render live video of the hands onto the display for a direct manipulation experience from afar.

Fast targeting to any point on the display: Touch-screen and tabletop display users can randomly access any point on the display by simply touching the desired location. As described earlier this is difficult to do when a separate touchpad surface is much smaller than the display to which it is directly mapped. We address this issue by using asymmetric two-handed input so that the dominant hand performs fine positioning towards a target while the non-dominant hand coarsely positions the space of the dominant hand.

Maximize comfort for from afar interaction: While our goal is to allow a user to interact with a large wall display while remaining seated at a table, we must still consider any potential discomforts that our interaction techniques may introduce. This includes allowing the user to adjust the position of the touchpad surface as well as minimizing awkward gestures.

Support for multiple concurrent users: In a conference room setting, it would be desirable to allow more than one user to access the display without affecting the work of others.

In addition to these design goals, we also consider the design issues outlined by Kjeldsen and Hartman [Kjel01] for vision-based user interfaces. In particular, our interaction techniques should consider the intuitiveness and learning curve required to perform a motion or gesture, the stability required by a user to perform a task, and the multiplexing ability offered to a user during the process of an operation.

4.3.3 System Overview

4.3.3.1 Display Hardware and Software

We use a 5m wide x 1.8m high rear-projection display consisting of a 3x6 projector array, where each projector is connected one-to-one with a 2GHz Pentium4 computer running at a desktop resolution of 1024x768 pixels. Using the open source Chromium library [Hump02], any standard OpenGL application can be distributed onto the projector array so that the projectors act as one single large display of up to 6144x2304 pixels.

4.3.3.2 Touchpad Tracking

Our hand tracking system is based upon the Visual Touchpad (VTP) device described in Chapter 3, which allows two unmarked hands to be tracked over top of a black rectangular surface using two off-the-shelf web cameras placed above the work area. For this system we used a simple piece of cardboard with a 60 x 20cm black region that resembled the shape and aspect ratio of our large screen.

As described earlier, the major advantage of the VTP over other touch-sensitive devices is the ability to extract the entire 2D image of each hand, which allows for differentiating between fingers. Additionally, the actual hand images can be extracted and rendered independently onto the screen as a visual proxy of a user's actual hands, providing richer feedback than a standard mouse cursor or even a virtual hand.

One problem with the original VTP was its requirement that the camera positions be fixed with respect to the touchpad surface. This limits the mobility of the device, and also prevents the detection of multiple devices/users from a single camera pair. In order to facilitate the use of multiple VTPs as well as make the device somewhat mobile while on a desk, we use the

ARTag library [Fial04] which allows up to 2048 unique 2D identifiers to be detected quickly and accurately in our captured camera images. By attaching such tags above the black rectangular region on each VTP (Figure 4.8), we can uniquely identify a large number of users. Additionally, the tag detection allows us to localize the position of the black rectangular region quickly, allowing the entire touchpad to be moved while the cameras remain fixed. This allows users to position the touchpad comfortably during interactions, supporting our third design goal.

The tracking system runs on a 2GHz Pentium4 computer, which provides enough processing power for tracking two touchpads/users quickly (<50ms per frame) using two web cameras per touchpad, each capturing at a resolution of 320x240 pixels.

4.3.3.3 Postures and Gestures

Figure 4.9 shows the set of static postures and temporal gestures that our system can infer. Note that each of these gestures can be overloaded based on whether or not a particular fingertip is making contact with the touchpad surface, or is tapping/double-tapping the surface.

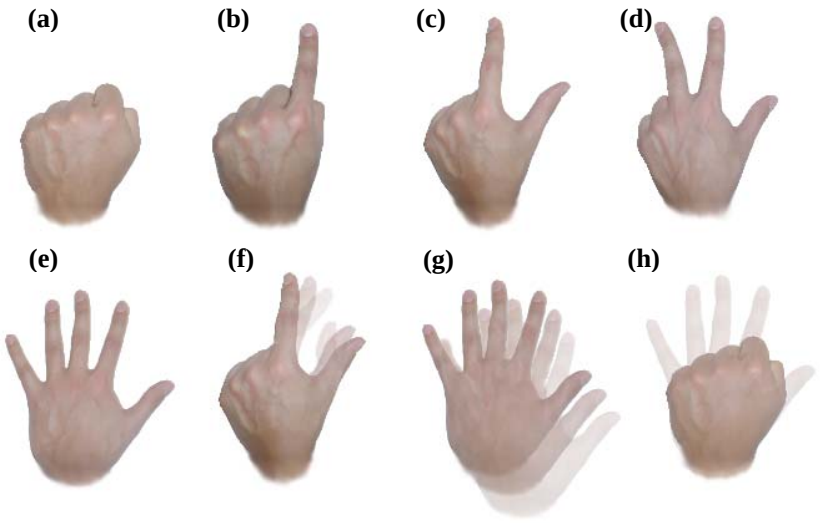


Figure 4.9 – Postures and gestures recognized by our system. (a) Fist posture; (b) Pointing posture; (c) double-point posture; (d) triple-point posture; (e) five finger posture; (f) pinching posture; (g) five-finger slide gesture; (h) grabbing gesture.

4.3.4 Interaction Techniques

In the following sections we describe the bimanual interaction techniques that we have developed for fluidly interacting with large wall displays from afar. Without loss of generality, we assume that a user's right hand is the dominant hand while the left is the non-dominant hand.

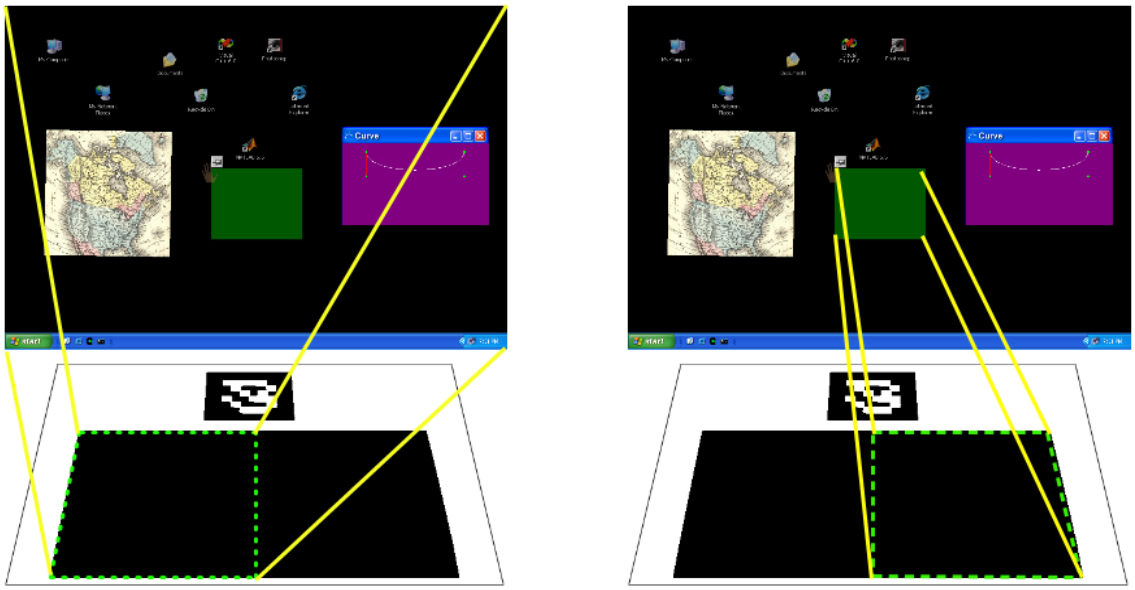
4.3.4.1 Coarse Positioning

The original Visual Touchpad directly mapped the touchpad coordinates into absolute display coordinates. This causes serious problems when attempting fine positioning tasks on large wall displays, since a small amount of movement on the touchpad gets mapped to a large number of display pixels. Since allowing fast and accurate access to all parts of the screen is a fundamental issue in large display interaction, we have developed an asymmetric two-handed technique to address this problem.

Asymmetric-dependent tasks, as proposed by Guiard [Guia87], are those in which the dominant hand moves within a frame of reference that has been set by the non-dominant hand. In other words, the non-dominant hand can be engaged in coarse and less frequent actions, while the dominant hand will be used for faster, more frequent actions that require precision. It has been shown that such asymmetric-dependent tasks lead to the best performance due to their resemblance to the bimanual tasks humans perform in the real world [Hinc97, Kabb94].

Since the VTP can differentiate between the left and right hands, we are able to map the touchpad to the display differently for each hand. In asymmetric mode the left half of the touchpad is mapped to the four corners of the entire display (Figure 4.10a). Therefore, when the user makes a pointing gesture with the left hand index finger and touches the tip onto the touchpad surface, the corresponding position in display space is computed and the segmented video image of the left hand is instantly moved to that location. A panning icon also appears at the left index fingertip to denote that the finger can also be moved along the surface of the touchpad for smooth panning (Figure 4.15a). While this allows random access to almost any part of the display similar to a touch-screen, fine positioning is difficult due to the resolution

differences between the touchpad and the display. In other words, mapping half of the 60cm width of our touchpad to the entire 5m width of the display means that even a 1cm change in the fingertip position results in a 16cm jump on the display. Additionally, our cameras introduce further inaccuracies depending on the capture resolution of the cameras (we currently capture at a resolution of 320x240).



(a) (b)

Figure 4.10 – Touchpad mapping for asymmetric interactions for: (a) the left hand; (b) the right hand.

4.3.4.2 Workspaces and Fine Positioning

Following Guiard's asymmetric-dependent principles, we place a green-colored, semi-transparent, rectangular workspace at the left index finger position, with the right hand rendered inside of this workspace (Figure 4.11). Thus the right hand can be used to perform more accurate positioning and manipulation tasks *inside* of this workspace, while the left hand coarsely positions the entire workspace anywhere on the display. For such right hand interactions, the right half of the touchpad is mapped to the four corners of the workspace (Figure 4.10b). This configuration minimizes any interference that may occur if the hands begin to overlap.

Using this combination of two-handed coarse and fine positioning, a user can quickly access any part of the display with ease, which supports our second design goal.

4.3.4.3 Selecting/Moving/Rotating Single Objects

Kjeldsen and Hartman [Kjel01] suggest that direct pointing, control, and selection tasks are well-suited to vision-based hand tracking interfaces due to their low learning curve compared to systems based on complex gesture sets. We leverage this knowledge for the purpose of manipulating objects in our system.

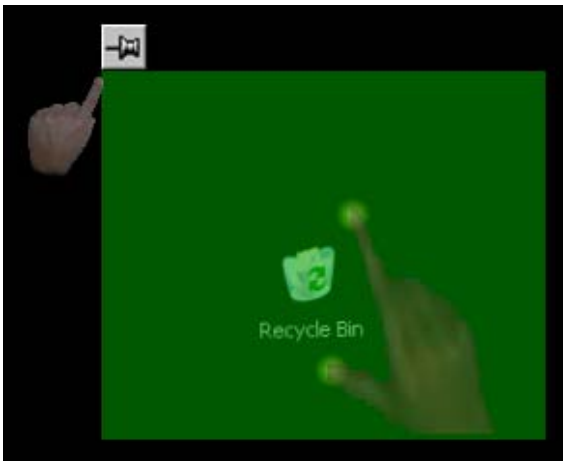


Figure 4.11 – A workspace that can be coarsely positioned using the left hand (shown at top-left of the semi-transparent overlay), while the right hand performs fine manipulations inside of it.

To select an object inside of the workspace, a pointing gesture is made with the right index finger. When contact is made with the touchpad surface, any object underneath the on-screen fingertip becomes selected. In effect, the right hand in a pointing gesture can perform any operation that a single-button mouse could perform, where clicking is simulated by making contact with the touchpad surface. With the right hand, any selected object can then be moved locally within the workspace by simply moving the finger across the surface of the touchpad. This allows for precise positioning of the object. Additionally, objects can also be rotated if desired by using the finger orientation information provided by our tracking system, as described in Chapter 3.

To quickly move selected objects to areas outside of the workspace, the user can hold an object with the right index finger while the left index finger is used to move the position of the workspace as described earlier. The selected object will remain attached to the right index finger and thus remains within the workspace as it moves, thereby allowing the object to be coarsely placed anywhere on the screen quickly, but without interfering with any precision

movements being carried out by the right hand. In other words, the right hand does not have to transition between coarse and fine positioning as might be required in single hand techniques for large display interaction. Figure 4.12 illustrates this interaction.

4.3.4.4 Selecting Multiple Objects

In traditional graphical interfaces, selecting multiple objects such as icons usually requires dragging a box around a group of objects using a mouse button. For multiple random selections, however, a user is typically required to use a modifier key on the keyboard to individually select each desired object. While we can simulate such selections using a second finger as a modifier, we propose an alternative approach that leverages the high degree of freedom input provided by our tracker. By making a five-finger grabbing gesture with the right hand as shown in Figure 4.13, the object closest to the centre of the palm of the hand is "grabbed" and disappears from the workspace. To help visualize which object will be grabbed, a line is drawn from the centre of the hand to the closest object. Repeating this for a number of objects, a large number of randomly placed objects can be selected quickly and precisely.

As multiple objects are grabbed, they are placed in a last-in first-out queue at each of the fingertips starting from the thumb and progressing in order to the little finger. To place these objects back into the workspace the user can make and hold a five-finger gesture above the touchpad surface. When this is done, the objects assigned to each of the fingers are displayed over top of the hand image on-screen, in LIFO order from the tip (Figure 4.14). Therefore, by tapping one or more fingers onto the touchpad, the object closest to the tip of the tapped finger(s) will be placed back into the workspace at the tapped location. With five fingers a user can easily grab up to 15 objects without cluttering the display using our system. However, this will vary based on the size at which icons are rendered.

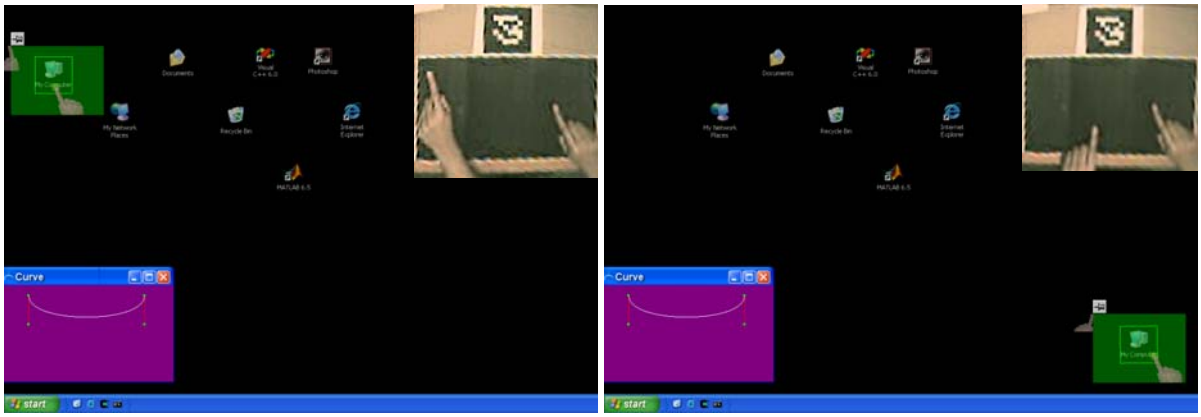


Figure 4.12 – An example of fast object movement using two-hands. The icon at the top left of the display is coarsely but quickly moved to the bottom right.

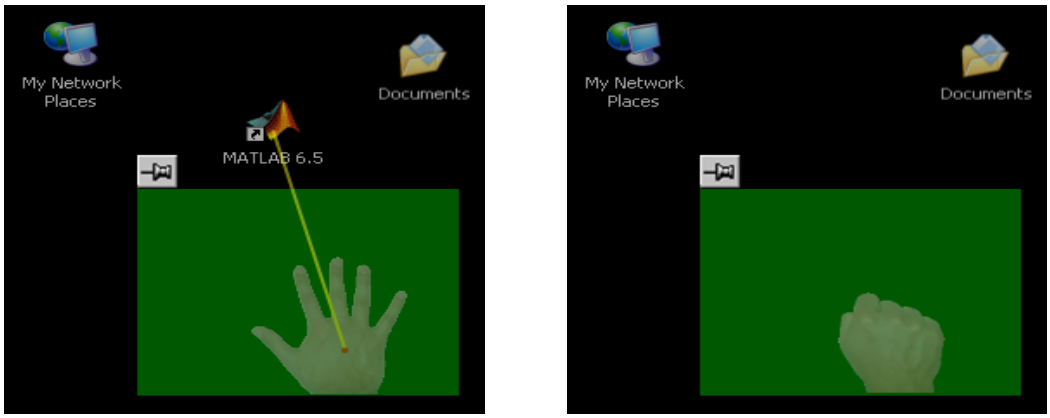


Figure 4.13 – Grabbing the object closest to the hand.

Figure 4.14 – Placing multiply selected objects. The selected icons appear on each finger based on selection order. Tapping a finger releases the icon closest to the tip.

4.3.4.5 Resizing/Zooming/Rotating Workspaces

By default, the workspace is set to a size such that every pixel on the display can be reached using a combination of coarse and fine positioning. However, since the right hand operates in a space where the right half of the touchpad is mapped to the corners of the workspace, the user is limited to a resolution of a single pixel. For precise object positioning this is ideal, but in some instances it might be desirable to work at a different resolution with the right hand.

To facilitate such instances, the left hand can be used to modify properties of the workspace. To resize the workspace, the user makes a pinching gesture with the left hand. A resizing widget then appears between the thumb and index finger of the on-screen representation of the hand to signify that a resize can be performed (Figure 4.15b). By increasing the distance between the two fingers, the workspace grows in both the horizontal and vertical directions (up to some predefined maximum size). Similarly, decreasing the distance between the fingers causes the workspace to shrink (down to some minimum size).

Figure 4.15 – Widgets drawn beside the on-screen representation of the hand for modifying workspaces: (a) Panning; (b) Resizing; (b) Zooming; (d) Rotation.

Increasing the workspace size reduces the resolution at which the right hand operates, while decreasing the size increases the resolution. To counter the effect of a resize operation, the user can also modify the zoom level of the workspace. By placing the left hand in a triple-pinch-posture with all fingers touching the touchpad surface, a zoom lens widget appears between the left hand's thumb and index finger (Figure 4.15c). Raising the left index finger off the surface then causes a non-linear zoom-in of the workspace towards the center, where the speed of the zoom depends upon the amount of time the finger is held above the surface. Similarly, raising the left thumb instead of the index finger causes a non-linear zoom-out to be performed. By zooming out to a level below the default zoom setting, the workspace can

provide a dollhouse [Swam97] view of the entire display contents. This allows for fast access to any item on the screen, albeit in a smaller form, which can be useful in certain situations.

Finally, workspaces can also be rotated by extracting the left index finger orientation during a pointing posture held above the surface. We assume that if the finger is generally pointing in the vertical direction of the touchpad, no rotation should be performed. However, if the direction falls below -10 degrees then the workspace begins to rotate in the counter-clockwise direction. Similarly if the finger direction is above +20 degrees the workspace rotates in the clockwise direction. In both cases, a rotation dial appears at the tip of the left index finger to signify the mode change (Figure 4.15d). This allows workspaces to be positioned with the left hand as one would adjust a piece of paper in real life before writing on it. This allows a user to better position the right hand in order to more precisely manipulate an object. To avoid awkward orientations, however, we limit the amount of workspace rotation to +/- 45 degrees from the vertical direction. Note that we chose unbalanced rotation thresholds since the left index finger generally points in the +5 degree direction from the vertical during typical touchpad usage. These should be reversed for left handed users.

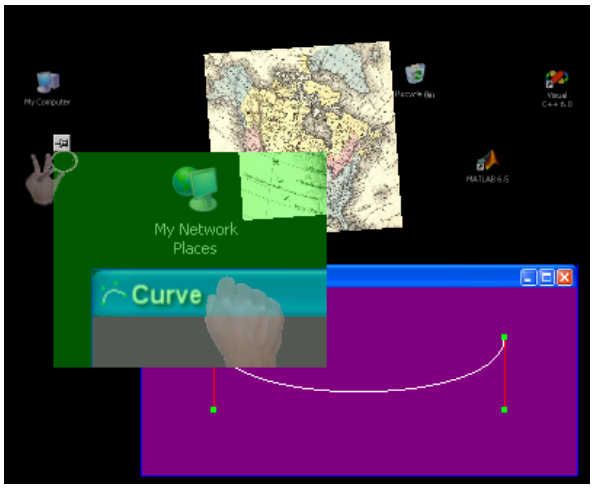


Figure 4.16 – Zooming a workspace using three-fingers.

Subsequent movements of the workspace maintain the size, zoom level, and rotation settings that have been set, thereby mimicking the functionality of magic lenses as proposed by [Bier93]. By combining resizing, zooming, and rotation operations, a user can work on

different parts of the display with the desired amount of visual feedback and positioning resolution (Figure 4.16). These operations support our design goal of maintaining comfort for the user, since they allow the user to place a workspace and the right hand into a maximally efficient pose.

4.3.4.6 Pinned Workspaces

In many large display applications, a user may need to frequently move between two (or more) completely different regions of the screen. If the user desires working in each of these regions at different granularities, this would require constant zooming and resizing operations after each move. To remedy this problem, we allow workspaces to be pinned so that their position, size, and zoom setting are locked. To do this a user makes a double tap gesture with the left index finger in a pointing posture. This toggles the workspace to pinned mode, causing the right hand to become locked inside of the pinned workspace. An icon at the top-left of the workspace depicts the pinned/unpinned state of the workspace. The previously described interactions can then be performed inside of this pinned workspace as usual. If the left hand is again placed in a pointing posture, a transparent "ghost" workspace is shown emanating from the left index finger position. As the left index finger is moved further away from the top-left of the pinned workspace, the ghost workspace becomes more opaque up until the overlap between the ghost workspace and the pinned workspace falls below 25%. At this point, the ghost workspace becomes the active workspace, and the right hand smoothly transitions into the active workspace. The pinned workspace remains at its original location, but right hand operations can now be performed inside of the active workspace as before. The active workspace can then be pinned elsewhere to create other pinned workspaces. If the active workspace is brought back towards a previously pinned workspace, and the overlap is greater than 25%, the active workspace becomes a ghost workspace once again and the right hand transitions into the pinned workspace (Figure 4.17). In this manner, a user can quickly move between different parts of a large display without worrying about size or zoom settings. Additionally, single or multiple object selections can also be made between pinned workspaces. To delete a pinned workspace, the user can simply move into the workspace's area and then double tap with the left index finger. This removes the pinned workspace, and the ghost workspace becomes the active workspace. The concept of multiple workspaces combined with the asymmetric movement techniques further supports our second design goal

of allowing fast access and targeting to all parts of the display, while simultaneously achieving our first design goal of leveraging both hands and multiple fingers effectively.

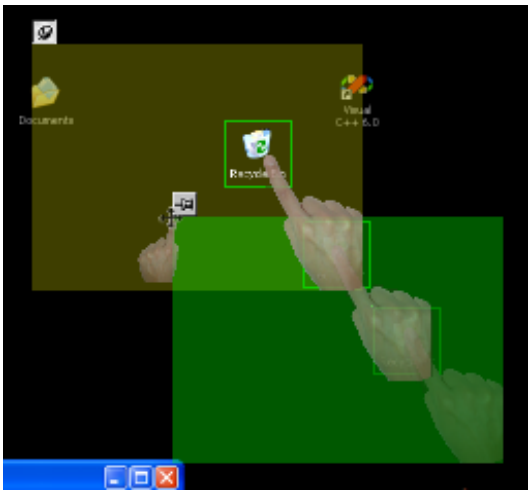


Figure 4.17 – Transitioning into a pinned workspace. The hand smoothly transitions into the pinned workspace, taking any selected objects along with it.

4.3.4.7 Facilitating Symmetric Bimanual Input

For certain tasks, a user may want to perform symmetric bimanual manipulations where both hands perform very similar functions in synergy. By default, the system supports asymmetric interactions, where the left hand is rendered at the top left of the active workspace as a small multi-point cursor. To perform fine operations with the left hand in a manner similar to the right, the user first makes a five-finger sliding gesture (Figure 4.9g) towards the bottom-right corner of the touchpad. This causes the left hand to smoothly transition *into* the workspace so that the mapping for both the left and right hands is such that the four corners of the touchpad correspond to the four corners of the workspace (Figure 4.18). To transition back to asymmetric interaction a five-finger sliding gesture is again made with the left hand, but towards the top-left of the touchpad.

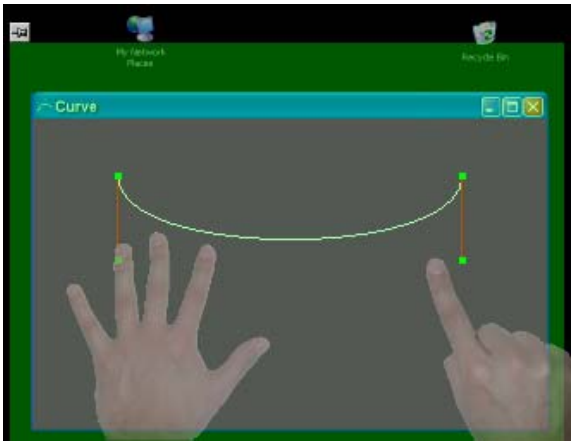


Figure 4.18 – Mapping both hands into the same workspace for symmetric bimanual tasks.

4.3.5 Discussion

This system design investigated a number of techniques for interacting with large displays from afar using a vision-based hand and touchpad tracking system. By allowing users to sit comfortably at a table in front of a large high-resolution display, traditional selection and navigation techniques become inefficient and other more appropriate methods must be developed. We presented a set of such approaches that leverage people's natural abilities to manipulate real-world items with their hands asymmetrically. Our current design satisfies our original design principles of: (1) leveraging two hands and multiple fingers for both natural and high degree of freedom input, (2) allowing fast targeting to any part of the display, (3) maximizing comfort for from afar interactions, and (4) supporting multiple users.

Although we have not yet performed a formal evaluation of our interaction techniques, a number of graduate students in our research lab were asked to try the system in order to gauge some early feedback on its strengths and weaknesses. Each user was first given a 5 minute introduction to the interaction techniques, followed by 10 to 15 minutes of supervised experimentation time where they were asked to perform simple manipulations.

All users were quick to point out that the basic movement and selection techniques were very intuitive, largely due to each user's familiarity with touch-surfaces, tabletop displays, and/or tablets. Additionally, every user found the rendering of the hands on the display (along with

the appropriate widgets and overlays) to be very compelling as well as informative, more so than the cursors typically used in large display interaction.

The use of the left index finger for coarse positioning of the workspace was found to be very intuitive by all users. However, some users felt that the default precision at which the right hand could manipulate objects was too coarse, thus requiring them to either reduce the size of the workspace or increase the zoom. This could be remedied by either using higher resolution cameras for the hand tracker or by moving the cameras closer to the touchpad surface. However, increasing the resolution would also increase the processing time as well as introduce noticeable lag on current CPUs.

While the workspace resizing gesture was found to be conceptually easily understood, one user complained that the three-finger gestures for zooming in and out were difficult and that the two-finger pinching gesture would be preferred for zooming. Unfortunately this would lead to an ambiguity with the current resizing gesture. Interestingly, Balakrishnan and MacKenzie [Bala97] showed that a pinching posture where the thumb and index finger work together provides a higher bandwidth input than using a single index finger. In a similar manner, it would be useful to determine what input bandwidth could be had from the three-finger gesture, since this could allow us to optimize the gesture for other more suitable operations.

The multi-point grabbing gesture was well received by all users, but the queue-based placement gesture received mixed reviews. Many users found that placing objects precisely with the ring finger and little finger was difficult since both of these fingers are difficult to control independently from one another. As a result, attempting to place an object from one finger would sometimes inadvertently also place the object from the other finger. This leads us to believe that these two fingers should not be used for independent precision tasks, but rather as a group modifier for the remaining fingers' tasks. This, however, needs further analysis to be confirmed. Another problem users had with the placement gesture was the queue arrangement. Users felt that they shouldn't be required to think ahead about the order of object placement during the grabbing phase, which the LIFO queue forced them to do.

One interesting suggestion was to allow a user to use their left hand to rearrange the ordering of objects in each finger.

While our interaction techniques currently don't provide any special support for collaborative tasks, an interesting side effect of using pinned workspaces in our system is their automatic support for multiple concurrent users. By pinning a workspace a user is effectively asking for exclusive access to a portion of the display. Therefore we can restrict other users from accessing a pinned lens that already has a user inside of it in a manner that is conceptually similar to the "carved" regions described in the Dynamo system [Izad03]. This further supports our design goal of allowing multiple concurrent users to interact without interference on the same display.

One unexpected feature of our transparent workspaces is their automatic "spotlight" functionality [Khan05]. Using a combination of workspace positioning with the left hand, pointing with the right hand, and speaking out loud, users could easily sway the attention of a small audience to a certain part of the large display extremely quickly. We plan to leverage this feature in the future more directly.

Finally, our current setup places two 320x240 cameras above the work area so that two touchpads can be detected accurately. Since our hand tracker requires a large amount of processing time, we have found that detecting more than two users seriously affects both speed and tracking accuracy. To detect more users we suggest adding extra machines and camera pairs, and then exchanging hand position information with a central node that manages a shared application. However, as processing power continues to increase, a single machine will be able to handle more than two users as well as higher resolution cameras.

In the future, we would like to investigate how to further integrate multiple users onto a large display using our system. In particular, with the high degree of freedom input provided by two hands and multiple fingers, it would be interesting to investigate what sort of collaborative tasks could be performed by two or more users working together. Another fruitful direction for research might be to investigate how vision algorithms could be further

leveraged for tasks other than just detecting hands. In a manner similar to the DigitalDesk [Well93], we could very easily place other objects onto the touchpad surface such as documents or other tangible objects, and then project them onto the large display. This opens up the possibility of using real tools to perform virtual tasks in more natural ways.

Another potential area for future research would be to investigate how users might transition between from-afar and up-close manipulations. Assuming that the large display has multi-touch capability, one possibility is to allow a *virtual* Visual Touchpad to be activated when the user directly touches the display. Therefore, up-close manipulations could be performed that still adhere to our proposed asymmetric interaction style, so that even unreachable sections of the display could be quickly accessed. We also imagine allowing fast transitioning between direct-touch and asymmetric states in the up-close scenario by simply using the five-finger sliding gesture that we currently use for symmetric interactions.

Finally, it would be interesting to investigate how up-close manipulations might be performed using a hand-held version of the Visual Touchpad, so that users could perform manipulations while potentially facing an audience. As discussed in Chapter 3, the use of two cameras mounted above the touch area makes it difficult to design a compact hand-held device based on the original Visual Touchpad. This suggests that an array of cameras mounted below the touch surface would be more appropriate, similar to the HoloWall [Mats97] or TouchLight [Wils04] designs. In such hand-held device scenarios, it would also be important to investigate variations of our existing asymmetric techniques, since the non-dominant hand would primarily be dedicated to holding the device which limits its use for direct manipulation operations.

4.4 Deaf Culture Centre Interactive Art Installation

The Deaf Culture Centre (Toronto, Canada) opened in May 2006 to highlight the contributions of the Canadian Deaf community. The Centre features an art gallery, museum, and gift shop which serve to highlight and archive Deaf historical artefacts and literature. The Centre also offers regular instruction in American Sign Language (ASL), which is a language

used by the Deaf community where thoughts are expressed using a combination of hand shapes and orientations, movements of the arms and body, and facial expressions.

The design of the Deaf Culture Centre's logo and branding was heavily influenced by some early work by Loomis et al. [Loom83], which reconstructed hand movements in three dimensions using computer graphics. While these reconstructions were primarily for analyzing movements in ASL, the resulting 3D trajectories could also be considered highly artistic. In a similar manner, the Centre used the Vicon optical tracking system to track the movements of one and two hands as they traced out various words in ASL. The raw tracking data was then given to a 3D artist for post-processing, where lofted 3D shapes were created for each word. Figure 4.19 shows a sample of some of the final lofted 3D shapes, along with their corresponding English translations. Words such as "community", "culture", and "inspire" were traced out since they captured the primary vision and mandate of the Centre. As can be seen, the final shapes are not only aesthetically pleasing pieces of art, but they also convey deeper meanings to those familiar with ASL. These shapes are widely used by the Deaf Culture Centre in all of their publications, brochures, logos, and even T-shirts.

Community *Culture* *Inspire*

Figure 4.19 – Example lofted shapes using Vicon data (courtesy of the Deaf Culture Centre, Toronto, Canada).

As part of the Deaf Culture Centre's exhibits, it was decided that a permanent interactive installation that allowed visitors to create their own similar art using hand shapes and motions would be desirable. However, the overall process of capturing hand motions using

the Vicon was determined to be impractical, since it required special gloves outfitted with markers as well as a time-consuming post-processing step by an experienced 3D artist. The cost of the Vicon hardware was also outside of the budget constraints for the exhibit. The Visual Touchpad technology, however, seemed to be a viable low-cost alternative on which to base the interactive exhibit. In the following sections we will describe the design and implementation of this system.

4.4.1 Design Principles

In designing the interactive art installation, we considered the following major issues:
Cost: A limited budget was allocated to the hardware components and tracking technology used in the exhibit, which immediately discounted the use of the Vicon system.

Robustness: The system must work reliably in an environment with high traffic, where lighting and shadows may change throughout the day. Additionally, mechanical parts should be minimized or at least made inaccessible to visitors.

Rich Real-time Visualizations: Hand shapes should be tracked in real-time and colourful visuals should be generated which resemble the lofted 3D shapes that were created with the Vicon system.

Ease of Use: Visitors should be able to quickly understand and use the system, without requiring any special setup or calibration phase.

4.4.2 System Overview

To meet our first design goal, the Visual Touchpad technology was used as the basis for the hand tracking system since it can be used with low-cost, off-the-shelf web cameras and standard PCs.

Figure 4.20 shows the proposed layout of the exhibit. A standard Pentium4 PC running at 3.0GHz was housed inside of a locked cabinet. The top of the cabinet was covered with black felt and was used as the interaction surface. A single Logitech Quickcam 5000 camera was

connected to the PC and attached above the black surface, pointing straight down. We opted for a single camera since explicit depth detection was not required for the visualizations (see section 4.4.3). A 20" LCD display was mounted against the back wall of the exhibit. All wires were hidden behind the panels, and the keyboard and mouse were placed inside of the cabinet. This particular setup reduces the number of mechanical parts that are accessible to visitors, which partially meets our second design goal.

Since we did not want to explicitly mark a white rectangular outline on the black surface (as is required by the original Visual Touchpad), we modified the Visual Touchpad system so that an administrator could mark the active area explicitly within the software using the mouse. This was a one-time configuration step, with the marked region's coordinates being saved to a data file.

Since the Visual Touchpad requires sufficient illumination to effectively segment out the hand from the black background, the exhibit includes a fluorescent tube light mounted above the camera area. The light combined with the black surface reduces the effects of shadows and helps to achieve the robustness design goal. The light also serves to illuminate the panels of the exhibit which include a brief introduction to ASL and instructions on how to use the interactive installation.

A professional one-minute video sequence was also filmed in front of the display, where a user described how to use the system both in ASL as well as in English. The video was integrated into the software and set to loop after every two minutes of inactivity. This feature, along with the informative panels, facilitates our ease-of-use design goal.

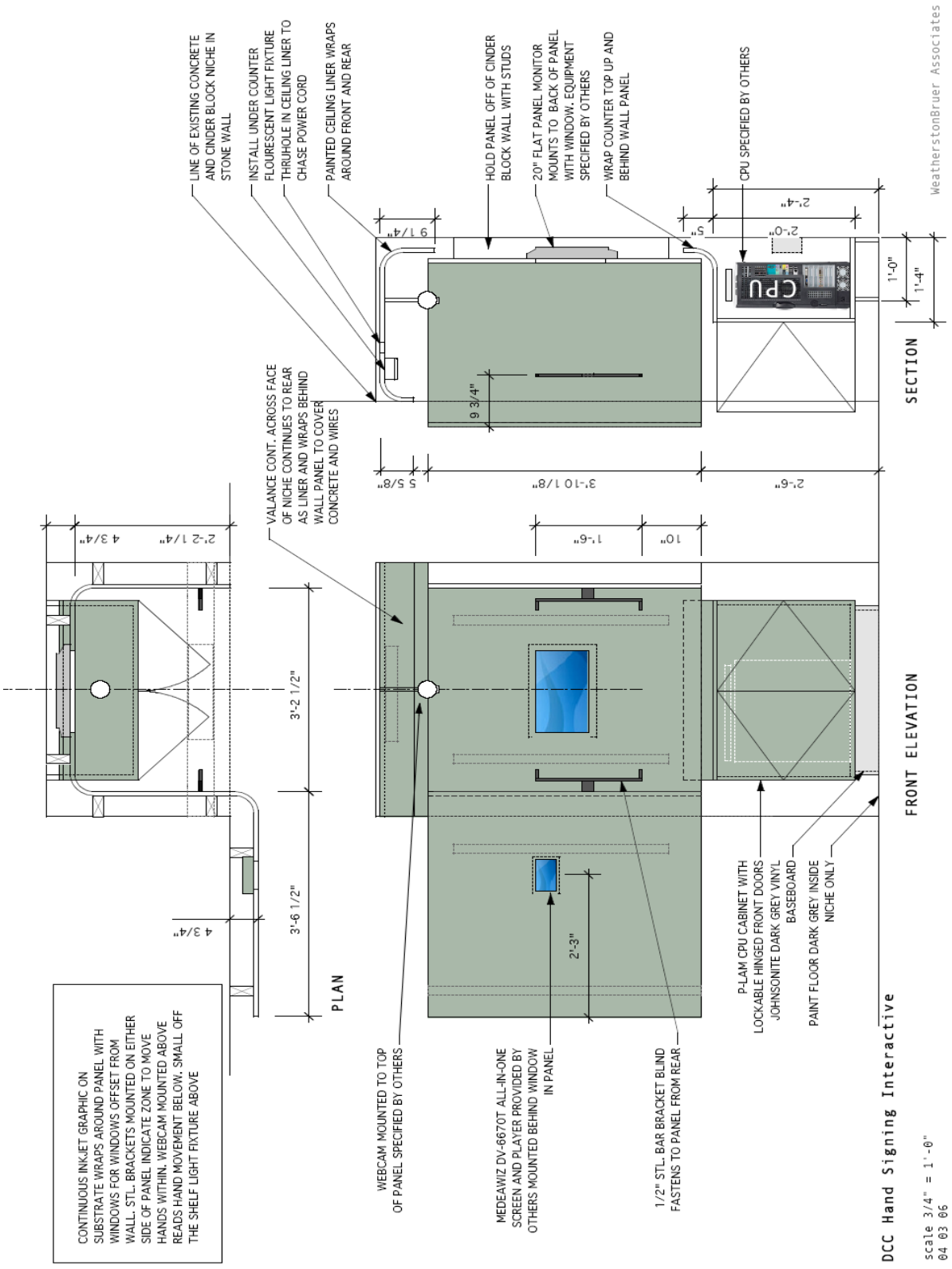


Figure 4.20 – Schematic diagram of the interactive installation (courtesy of WeatherstonBruer Associates).

4.4.3 Visualizations

Since the Visual Touchpad can provide us with hand contours in real-time, the visualization module generates lofted shapes by connecting the 2D hand contours from frame to frame to create continuous polygonal objects that appear to be 3D. As mentioned earlier, since we only use a single camera for the interactive exhibit, the Visual Touchpad only provides us with 2D position and orientation information for each fingertip. This is sufficient for our purposes since visualizations will rely on the hand contour to define the shape of the lofted surfaces instead of using explicit 3D information. For example, the mapping between the black surface and the display is absolute, so that movements to the left, right, forwards, and backwards in 3D space correspond to left, right, up, and down hand movements on the display. Similarly, up and down hand movements in 3D space will be interpreted as hand contour size changes from the camera's viewpoint, which will be reflected in any shapes which connect hand contours in a sequence of video frames.

Figure 4.21 outlines how contours are connected from frame to frame. At frame i, the contours for each detected hand are extracted from the Visual Touchpad. The system also keeps a history of hand contours from up to M previous frames (the current system uses M=60, which was based on a 20Hz average capture rate that roughly corresponds to 3 seconds of contour history). Let C_i represent a contour at frame i, let N_i represent the number of vertices in C_i, and let $V_{i,j}$ represent the j-th clockwise 2D point along contour C_i. The first phase of the visualization system involves establishing correspondences between contour points in adjacent frames. We currently use a simple approach where an initial correspondence between $V_{i\text{-}1,0}$ and $V_{i,0}$ is created. While an approach that establishes initial correspondences between actual fingertip locations would be more accurate, we have found the zero-index correspondence to provide acceptable results. We then create the remaining correspondences by interpolating contour point indices between C_i and $C_{i\text{-}1}$ based on the number of vertices in each contour. For example, if N_i is greater than or equal to $N_{i\text{-}1}$, then approximate correspondences can be set between $V_{i,a}$ and $V_{i\text{-}1,b}$ where $b=a*N_{i\text{-}1}/N_i$. If N_i is less than $N_{i\text{-}1}$, we set $a=b*N_i/N_{i\text{-}1}$.

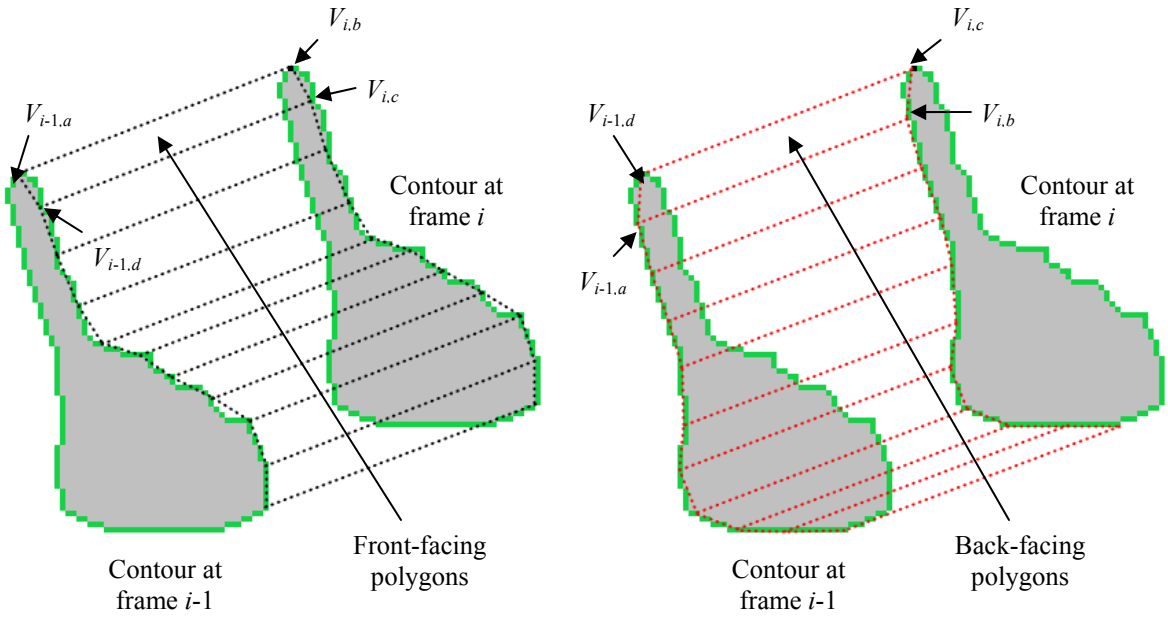


Figure 4.21 – Connecting hand contours from frame to frame.

Given all correspondences, we define four-sided polygons as ($V_{i-1,a}$ $V_{i,b}$ $V_{i,c}$ $V_{i-1,d}$) where a and b are corresponding indices and c and d are corresponding indices. Since the contour points are in clockwise order, d is the next sequential contour point index after a in C_{i-1}, and c is the next sequential contour point index after b in C_i, taking interpolation into consideration. Note that the 2D contour positions are based on the capture resolution of the camera (320x240). We therefore scale all contour coordinates into the resolution of the display (800x600) when creating polygons.

The next phase of the visualization system assigns colours to each polygon vertex. We first compute the mean position of the contours C_i and C_{i-1}, while the vector Q denotes the difference between the mean positions which approximately describes the direction in which the hand moved between the two frames. We then use Q to define ambient (r,g,b) colours for C_i as follows:

```
If Qx < 0:
      If Qy < 0:
            r = abs(Qx);
            g = abs(Qy);
            b = 0;
```

```
            Else:
                  r = abs(Qx);
                  g = 0;
                  b = abs(Qy);
      Else:
            r = 0;
            g = abs(Qx);
            b = abs(Qy);

      r = clamp(r);
      g = clamp(g);
      b = clamp(b);
```

This effectively assigns colours based on the speed and direction of hand movement, where speed controls intensity and direction controls the colour.

In order to provide more dramatic effects when fingers are outstretched, we increase the brightness of contour points that are within some threshold distance to a fingertip along the contour. Therefore, for a contour point $V_{i,j}$ that is f units from a fingertip in C_i, where D is our distance threshold and $f < D$, we increment the (r,g,b) components by $(D-f)/D$ and then clamp the values to 1. We currently use a value of 8 for D.

We then adjust the intensity of polygons that are back-facing as determined by their vertex order (see Figure 4.21). For back-facing polygons, all (r,g,b) components are scaled by 0.75. This creates a sense of depth since these darkened polygons appear to be in a shadow cast by the front-facing polygons.

Since we do not want to create infinitely long connected contours, we limit the length based on the contour history described earlier. Therefore, polygons are created between all adjacent contours from C_{i-M} to C_i, where i is the most current contour. In order to prevent abrupt ends, we adjust the intensity of polygon vertices based on the corresponding contour's history index. Therefore, the (r,g,b) components for a contour j are scaled by $1.0 - (i - j) / M$, where

i-M <= *j* <= *i,* resulting in shapes which appear to gradually fade out at the tail end over a few seconds (based on the value of *M*).

Finally, all vertices have an alpha transparency value of 0.75. This allows the shapes to appear partially translucent much like the lofted shapes generated with the Vicon data. The vertex data for all front and back facing polygons is then passed to OpenGL for rendering, where the (*r,g,b*) values as well as the alpha are interpolated across the pixels of each polygon, resulting in smooth gradations between each connected contour. Figure 4.22 shows some example shapes generated by the software.

It is important to note that, due to the simple correspondence approach, contour connections may occasionally result in self-intersecting polygons. This is particularly apparent when there is a change in the number of hands between two frames. For example, in the case where two hands make contact with one another, the system interprets both hands as a single large right hand. Therefore, contour connections for the misinterpreted right hand will be made with a correct right hand in the previous frame, which will likely result in poor contour point correspondences and thus a large number of self-intersecting polygons. While these self-intersecting polygons results in some visual anomalies, they do not adversely affect the interactive experience and in fact appear to be somewhat artistic.

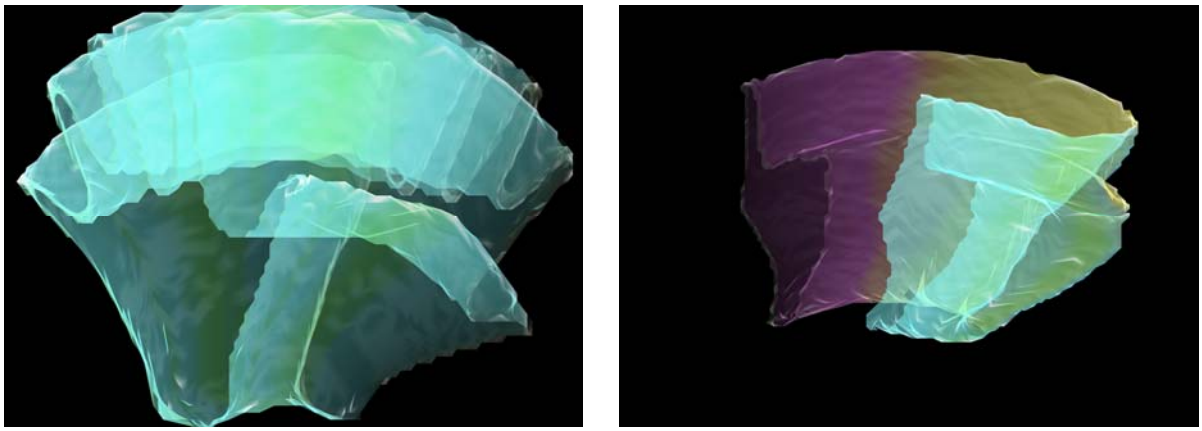


Figure 4.22 – Hand visualizations vary depending on the speed and direction of overall hand movement, and the number of fingers that are detected.

4.4.4 Discussion

Figure 4.23 shows images of the final installation at the Deaf Culture Centre. The system has been running continuously since May 2006 for 24 hours a day, seven days a week, with an occasional restart in the case of a power failure or if the software needs to be updated with a new version. The Centre is open to the public 6 days a week for approximately 8 hours each day, and weekly visitors average around 250, with occasionally larger groups during visits from schools.

Figure 4.23 – Images of the final installation at the Deaf Culture Centre in Toronto.

By relying on the Visual Touchpad technology, the system is immediately usable by all visitors and does not require any calibration for different skin tones or hand sizes. The lack of mechanical parts also eliminates the potential of wear and tear as well as reduces hygiene issues that are commonly associated with public installations. Additionally, the cost of the system is extremely low, consisting of a standard PC and a less than $100 web camera.

Feedback regarding the installation has been extremely positive, and it is often in high demand when school groups visit the Centre. Visitors experienced in ASL often create meaningful words and messages with the system, and a number of visitors have even inquired about the possibility of purchasing a print-out or DVD of their captured hand motions.

The first release of the software did not feature the instructional video due to production delays, and thus visitors were forced to rely on only the textual descriptions on the panels for usage information. This often resulted in visitors not fully comprehending the purpose of the exhibit, and they would often fail to place their hands into the active area, thereby missing

out on the interactive experience. However, when the exhibit was first demonstrated to them by a staff member, visitors would immediately grasp the purpose of the exhibit and would be much more comfortable using it on their own. By integrating the short instructional video into the system, staff members immediately noticed a significant increase in the number of visitors that used the interactive successfully without first being introduced to it.

4.5 Summary

This chapter presented the design and implementation of three systems that explored how multiple fingers could be used in various application scenarios. The first design, a simple picture manipulation application, served to highlight the capabilities of the Visual Touchpad for performing bimanual and multi-finger manipulations on a standard desktop PC. The second design investigated how multiple hands and fingers could be used to perform fluid manipulations on a large upright display from a distance. Finally, our third design demonstrated how the basic Visual Touchpad technology could be deployed into a robust real-world interactive art installation to showcase the expressiveness of hand shapes and motion from the perspective of the Deaf community. Taken together, these three system designs demonstrate the viability of using lightweight vision-based hand and finger tracking technology in real-world HCI scenarios while also showing how devices that can detect multiple fingers allow for expressive, high degree-of-freedom input.

Chapter 5

An Exploration and Evaluation of Bi-digital Input

5.1 Introduction

While many existing multi-point systems demonstrate using the index fingers of both hands for controlling bimanual interfaces [Reki02, Wils04, Wils05, Benk06], there has been very little investigation into how multiple fingers of a single hand can be used effectively. Such unimanual multi-finger techniques could be beneficial in situations where bimanual interactions are difficult or impossible to perform, such as when using portable hand-held devices where one hand is pre-occupied with holding the device. Similarly, utilizing multiple fingers from a single hand could be useful for enhancing the status quo single-finger interaction techniques, or to complement or enhance many bimanual techniques. Our large display interaction techniques from Chapter 4 found that users had difficulty using their middle, ring, and little fingers for manipulations, but had little to no difficulty with their thumb and index finger. This suggests that these two fingers may offer the most potential for facilitating high degree-of-freedom input with a single hand.

A few researchers have proposed interaction techniques that leverage the capabilities of the thumb and index finger of a single hand [Krue91, Igar05, Wu03]. In most cases, these multi-

finger techniques use these two fingers in a symmetric manner where each finger plays the same role, either in phase or out of phase. In this chapter, we formalize an interaction paradigm where the thumb and index finger of a user's dominant hand operate in an asymmetric-dependent manner to control *bi-digit widgets* (an example of which is shown in Figure 5.1). Based on results from the motor control literature, we argue that using the independent positioning capabilities of the thumb as a secondary parameter is a natural and expressive way to extend or support primary manipulations with the index finger. We explore the design space of such asymmetric two-fingered interactions by presenting a variety of widgets that use the relative distance or angle between the thumb and index finger to control a one-dimensional valuator in a discrete or continuous manner. We also support our bi-digital interaction style by presenting the results of a controlled experiment which compares performance differences when symmetric and asymmetric roles are assigned to the thumb and index finger during a compound selection task. Given that there is no standard interaction paradigm for multi-point touch-sensitive surfaces, we believe that our two-fingered design explorations and guidelines can potentially significantly influence future multi-finger user interface designs.

Figure 5.1 – The ThumbToolglass widget uses the thumb asymmetrically to rotate the menu, while the index finger is used to click-through the active menu option onto a target in the work area.

5.2 Related Work

Although many existing multi-point systems demonstrate some simple multi-finger interaction techniques to show the capabilities of the underlying hardware and software [Diet01, Reki02, Wils04, Wils05, Han05], it is still not clear how to best utilize the additional degrees of freedom offered by them. One of the earliest explorations into the possibilities of multi-point touch tablets was the work by Buxton et al. [Buxt85], which showed multiple fingers of both hands being used on a single touch-sensitive surface to adjust multiple one-dimensional parameters using virtual sliders. As discussed in Chapter 2, a popular use of multi-point surfaces is to map single finger movement to 2D cursor control, finger contact to mouse button clicking, and various static hand postures and temporal gestures to common graphical interface commands such as copy and paste [Reki02, Fing05]. Another popular approach is to use the index finger of each hand for performing bimanual operations in a symmetric or asymmetric manner [Krue91, Reki02]. Wu and Balakrishnan explored a variety of new interaction techniques that are possible with multiple fingertips of a single hand [Wu03]. Their multi-user RoomPlanner software for interactive tabletops showed that the distance between two fingers could be used as a continuous parameter for quickly rotating or scaling objects. Rekimoto [Reki02] also showed similar two-fingered techniques being used for zooming, translating, and rotating a virtual map. Finally, Igarashi et al. [Igar05] demonstrated an animation system whereby multiple fingertips could be used as constraints on a triangulated 2D object to smoothly animate and deform the geometry. Unfortunately, most of these proposed techniques for multi-point touch surfaces are somewhat arbitrary and do not consider actual human ability in performing the various manipulations.

One of the few comprehensive explorations into the use of the hand for computer input was the work by Sturman et al. [Stur89, Stur92]. They presented a taxonomy of hand motions for interaction in a virtual environment which leveraged the capabilities of a six degree-of-freedom glove that provided palm position and orientation, along with finger flex angles. Based on these input parameters and the hand motion taxonomy, they showed that the hand could be used effectively for simulating buttons (via postures and gestures), for controlling

valuators such as sliders or dials (via finger flex angles), and as a continuous 3D locator (using the 3D position of the palm).

In short, while there has been a significant amount of work in developing reliable multi-point touch surfaces in various form factors, there are no design guidelines on how to effectively utilize the increased degrees-of-freedom offered by multiple fingers of a single hand on these surfaces. Such information could be extremely useful for developing advanced multi-finger graphical widgets that may allow for more efficient interactions in the various domains where touch-sensitive surfaces are currently utilized.

5.3 Exploring the Design Space of Bi-digital Tasks

5.3.1 Motivation

Developing user interfaces that leverage multiple fingers is appealing for a number of reasons. Card et al. [Card91] suggest that input devices which use muscle groups having a large representation in the motor cortex have the potential to provide high performance, and the fingers of the human hand clearly fall into this category. The work by Zhai et al. [Zhai96] supports this idea by showing that completion times for a six degree-of-freedom docking task were significantly shorter when all the fingers of the hand were used as part of the manipulation. Similarly, Balakrishnan and MacKenzie [Bala97] found that the thumb and index finger working together (holding a stylus) outperformed the single index finger, wrist, and forearm in a pointing task.

From the perspective of touch-sensitive devices, these findings are important since the status quo of using a single index finger for manipulations only allows for two translational degrees-of-freedom, a contact state, and occasionally a hover or tracking state. While many pen-based interaction techniques are applicable to these single-point touch-sensitive devices, the high bandwidth capabilities of multiple fingers may allow for more expressive interactions in a device-free manner.

If we ignore current hardware limitations, an ideal multi-point touch-sensitive surface should provide us with the following information for each fingertip:

- User ID
- Hand (left, right)
- Label (thumb, index, middle, ring, little)
- Position, Orientation, Height above surface, and Pressure

Obviously, if we focus on single-user unimanual tasks, the various degrees of freedom for each finger (position, orientation, hover, pressure) are not completely independent. For example, movement of the ring finger typically causes unintentional movement of both the middle and little fingers due to the arrangement of the muscles in the hand [Hage00]. We observed such enslaving effects in our large display interaction techniques from Chapter 4. Similarly, crossing fingers over one another is extremely difficult, and the range of space where the fingers can be simultaneously positioned is very limited.

In the case of serial tasks such as touch-typing, independence is not a major problem since the degrees-of-freedom for each finger can be controlled one at a time. However, for many tasks it is desirable to maintain a primary focus while a secondary control is adjusted simultaneously. Without such facilities, users must frequently move back and forth between a work area and toolbars or system menus.

Bimanual interfaces have been shown to be quite effective in this regard [Benk06, Bier93, Kabb94, Mats00, Mott01]. An alternative approach, which hasn't been studied as extensively, is to leverage the independent positioning capabilities of multiple fingers from a single hand. This could be extremely useful in situations where the second hand is unavailable, such as when using hand-held devices, or to enhance existing bimanual interfaces. To facilitate more fluid interactions with a single hand, the thumb and index finger appear to be the two most interesting digits for the following reasons:

- Due to its opposability, the thumb has a much larger range of motion than the other fingers of the human hand [Mack94].

- The index finger is considered to be the most *dominant* digit of the hand in the general population when used for selecting single targets [Raj99].
- Under instructed movements with a single finger, the thumb and index finger exhibit the least amount of unintentional enslaved motion in the non-instructed digits (i.e. they offer the highest amount of independent control) [Hage00].
- Under two-finger force production tasks, the thumb and index finger combination results in the highest amount of individuation compared to all other two-finger combinations [Reil04].

The frequent use of the thumb and index finger in everyday electronics devices such as multi-button computer mice, PDAs, cellphones, and video game controllers further motivates the use of these digits on multi-point touch-sensitive surfaces.

5.3.2 Bi-digital Symmetric and Asymmetric Tasks

By focusing on two digits for our initial exploration, we are able to draw comparisons with well-studied techniques from the bimanual interface community. Indeed, if we use terminology similar to that used in the bimanual interaction literature, the majority of single-handed bi-digital techniques that have been proposed would fall into the *symmetric-dependent* category of interactions where each finger is assigned the same role, either in phase or out of phase, in order to complete a compound task. For example, Rekimoto's multi-finger map browsing tool [Reki02], when used with two fingers of a single hand, assigns each finger to act as a constraint on the underlying map location, allowing the map to be zoomed, panned, or rotated. Similarly, Igarashi et al.'s [Igar05] multi-finger shape manipulation system also treats each finger as a constraint in order to pan, rotate, and deform 2D objects. Moscovich and Hughes [Mosc06] also use a symmetric mapping to control an adjustable area cursor, where the midpoint between the thumb and index finger controls the position of a relative cursor while the span between the two fingers adjusts the size of the cursor. Other symmetric-dependent bi-digital tasks include Wu and Balakrishnan's parameter adjustment widget [Wu03] and Smart Technologies two-fingered "right-click" activation [Smar05].

The alternative class of bi-digital techniques based on an asymmetric mapping of the fingers has not been studied as extensively as the symmetric approaches. This is rather surprising considering that the bimanual interface literature has demonstrated a variety of interesting two-handed interaction techniques where asymmetric roles are assigned to each hand. Therefore, as an exploration of the design space of bi-digital input, and based on the various affordances of the thumb and index finger described earlier, we propose an asymmetric mapping of the fingers as follows:

- The index finger defines the focus of a manipulation and performs the primary tasks.
- The thumb performs secondary actions to modify or support the operations of the index finger.
- The thumb and index finger may perform actions either serially or in parallel.

In other words, the index finger can be used to set the focus of a manipulation since users are accustomed to using this finger for existing single-point touch-screen interactions. The thumb can then be assigned a secondary role to adjust properties that support the index finger's manipulations. We feel that investigating such *bi-digital asymmetric-dependent* techniques that assign separate but dependent roles to each finger is a promising direction for research since they may allow for fluid localized secondary interactions on touch-sensitive surfaces.

5.3.3 A Taxonomy of Bi-digital Tasks

To our knowledge, there are only three techniques that have used two or more fingers in an asymmetric manner. The first was the two-fingered toolglass presented by Wu and Balakrishnan [Wu03] which allowed one finger of the dominant hand to position a tool palette in the work area while another finger of the same hand was used to click-through the desired menu item. Similarly, although they used a marker-based 3D tracking system rather than a touch-sensitive surface, Vogel and Balakrishnan's ThumbTrigger [Voge05] was another asymmetric two-fingered technique since they used the index finger to position a cursor on a large display while pressing the thumb against the side of the hand simulated a mouse button click. Finally, Grossman et al.'s marker-based thumb scrub gesture allowed for continuously translating a 3D model along an axis defined by the index finger when interacting with a volumetric display [Gros04]. While each of these techniques individually

demonstrated their usefulness for the different tasks, they were proposed without an underlying model or framework that could be used to generalize them to other types of asymmetric tasks.

As a step in this direction, Table 5.1 formalizes the design space of single-handed bi-digital dependent tasks as a taxonomy, drawing on the existing techniques from the literature as well as our own proposed widget designs. Note that the taxonomy is general enough to encompass not only techniques for one planar touch-sensitive surface, but also devices which may have multiple touch sensors arranged in different configurations (such as two touch-sensitive surfaces mounted on the front and back of a hand-held device, or two surfaces orthogonal to one another).

Table 5.1 - Taxonomy of unimanual bi-digital dependent tasks on multi-point touch-sensitive surfaces. Our proposed techniques are displayed in italics.

	Dependent Tasks	
Parameter Control	**Symmetric**	**Asymmetric**
Continuous-Continuous	Two-finger map browsing [Reki02]. Two-finger shape manipulation [Igar05]. Two-finger parameter adjustment [Wu03]. Adjustable area cursor [Mosc06].	*ThumbSlider, ThumbWheel, ThumbTrack (with continuous index finger control).*
Continuous-Discrete	X	Two-finger toolglass [Wu03]. *ThumbSlider, ThumbWheel, ThumbTrack (with discrete index finger control).* *ThumbMenu, ThumbToolglass, ThumbSwitch (with continuous index finger control).*
Discrete-Discrete	Right mouse button activation [Smar05].	*ThumbMenu, ThumbToolglass, ThumbSwitch (with discrete index finger control).*

The taxonomy considers both symmetric and asymmetric dependent tasks, with each task further divided based on the type of parameter control being assigned to some particular degree of freedom of each finger. For parameter control we adopt the two types of actions as proposed by Sturman [Stur92]: *continuous control*, where a continuous quantity is derived from some degree of freedom of a finger based on the precision of the input device, and *discrete control*, where the continuous quantity is discretized into specific values or ranges.

Our taxonomy does not explicitly consider other aspects of finger motion such as direct manipulation vs. symbolic gestures or absolute vs. relative parameter control since these are already well defined by other taxonomies [Hinc99, Stur89, Stur92].

Finally, we do not currently consider the class of *bi-digital-independent* tasks, where each finger is assigned a distinct, independent role. Such manipulations are analogous to "tapping the head while rubbing the stomach" as described by Kabbash et al. [Kabb94], and since these particular types of techniques do not appear to work well in the bimanual case leads us to believe that they will be even less useful for bi-digital tasks where the fingers already have a limited amount of independence.

5.4 Asymmetric Bi-digit Widget Designs

Assisted by our taxonomy, we designed a set of general-purpose asymmetric bi-digital widgets which allow for localized control of a secondary parameter in a continuous or discrete manner. We currently assume interactions will be performed with the right hand, but the various widgets are easily modified to accommodate left-handed use.

Since there are a large number of ways in which we can combine the various degrees-of-freedom for the thumb and index finger, we have decided to concentrate on techniques which map the relative distance or angle between the thumb and index finger to a one-dimensional valuator. This particular degree-of-freedom is interesting since the hand can be placed comfortably on a touch-sensitive surface with the standard pointing gesture for primary index finger manipulations, while the outstretched thumb still has a sufficient range of independent motion for performing secondary manipulations (Figure 5.2). Therefore the index finger can be used in a familiar manner to select and manipulate objects, while the thumb can be outstretched when desired in order to activate a desired widget and perform appropriate subtasks. This allows for smoothly merging command or parameter adjustment with direct manipulation, but without requiring the index finger to change focus to invoke the operation as is required with single-finger techniques such as FlowMenus [Guim00].

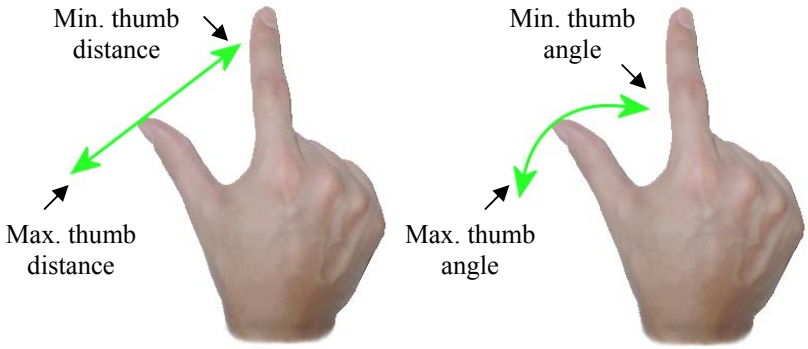


Figure 5.2 - Using the relative distance or angle between the thumb and index finger as a valuator.

5.4.1 Enabling Technology

We use the vision-based Visual Touchpad (Chapter 3) as a test bed for rapidly prototyping our bi-digital widgets, which allows us to interact on a horizontal surface while visualizations are depicted on a standard upright display. The advantage of the Visual Touchpad is its ability to extract the label, tip position, orientation, and hover information for any outstretched finger located above the touch surface. Our bi-digital widgets make use of this information to determine the operation of each finger. On many existing multi-point touch-sensitive surfaces, however, it is difficult to determine finger labels due to the lack of a reliable 2D image of the hand. To overcome this limitation, finger labels for two fingers can be simulated by assuming that the first contact point is the index finger, while a second contact point is the thumb [Wu03]. Additionally, for devices that do not directly detect finger hover, the SimPress technique proposed by Benko et al. [Benk06] can be used as an effective approximation. Finally, a calibration phase is required for each user in order to define the most comfortable minimum and maximum thumb distance or angle which can then be used to constrain the range of the valuator. In the remaining descriptions we assume that the relative angle between the thumb and index finger is used to control the valuator, but thumb distance can be used just as effectively for multi-point devices that only detect fingertip positions.

5.4.2 Continuous Bi-digit Widget Designs

5.4.2.1 ThumbSlider

The ThumbSlider widget maps the minimum and maximum relative angle of the thumb to a continuous value between 0 and 1, which effectively allows the thumb angle to be used as a slider with an absolute mapping. Since the slider is adjusted only when the thumb makes contact with the touch-sensitive surface, the current value can be locked by simply raising the thumb above the touch surface and then hiding it in the palm of the hand. Additionally, since the index finger is free to perform primary discrete or continuous manipulations such as selecting or manipulating objects, the thumb can be used to simultaneously adjust a secondary parameter for more sophisticated interactions. This sort of continuous control may be useful for localized subtasks such as smoothly zooming the canvas at the index finger location, resizing the drawing tip in a paint program, or scrolling an active document in one dimension. We also imagine this absolute control being used as an alternative to Moscovich and Hughes adjustable area cursor [Mosc06], since the position of the cursor can be specified more explicitly with the index finger using our approach. Figure 5.3 shows the ThumbSlider being used as a simple secondary continuous control, while the index finger controls the position of the standard arrow cursor. The visualization of the ThumbSlider is relative to the cursor position so that it only moves when the cursor moves. The thumb angle therefore only modifies the one-dimensional position of the marker on the ThumbSlider's track.

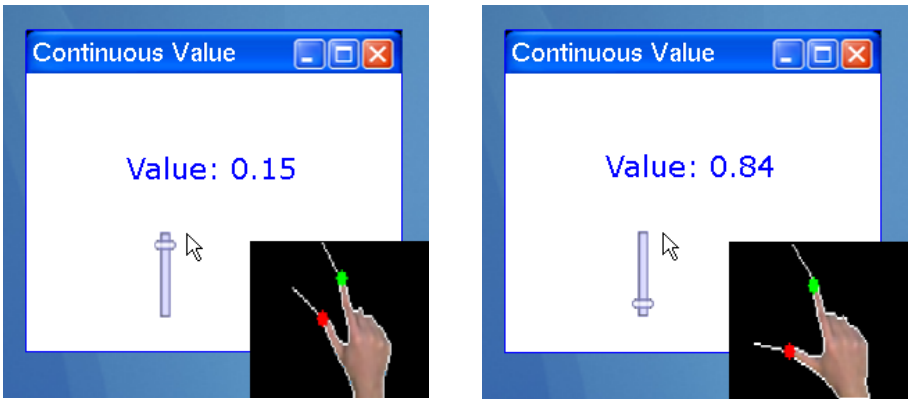


Figure 5.3 - The ThumbSlider widget uses the thumb angle as a secondary absolute continuous valuator.

5.4.2.2 ThumbWheel

The main advantage of the ThumbSlider is that it is very easy to select a value due to the absolute mapping. Unfortunately, the accuracy of the continuous control is dependent upon the precision of the touch-sensitive surface as well as the limited range of thumb motion. Therefore it becomes very difficult to make fine parameter adjustments as well traverse a large range of values with one ThumbSlider widget. Our ThumbWheel is an alternative continuous control which combines relative changes of the thumb angle with an acceleration function to adjust the continuous parameter. Figure 5.4 shows the motion that is used to control the ThumbWheel, along with the widget's visualization. To increase the continuous parameter value, the user makes a flicking motion with the thumb in a direction towards the index finger. Similarly, the parameter can be decreased by making a flicking motion away from the index finger. This particular widget therefore provides the functionality of a mouse wheel on touch-sensitive surfaces, but with speed-dependent behaviour that more closely resembles a trackball. By using a one-dimensional adaptation of the Windows XP pointer ballistics algorithm [Micr05], the ThumbWheel allows for both precise parameter adjustment as well as traversal of a large parameter range, but with the added cost of a clutching mechanism. Clutching is possible since we only increment or decrement the parameter value when the thumb makes contact with the touch-sensitive surface. Similar to the ThumbSlider, the ThumbWheel allows the index finger to perform primary 2D operations such as setting the focus of the manipulation while the thumb controls a secondary parameter to support the index finger's manipulation.

Figure 5.4 - Flicking motion for the ThumbWheel widget. Left flicking decreases a continuous valuator, while right flicking increases the valuator. The speed of the flick controls the granularity of the parameter adjustment. The widget is visualized with a rotating dial.

5.4.2.3 ThumbTrack

Although the ThumbWheel resolves some of the limitations associated with the ThumbSlider, repetitive clutching is required in order to traverse a large parameter range which might be fatiguing. For tasks which require such large traversals frequently, the ThumbTrack widget allows continuous parameter control in a manner similar to an isometric joystick or track-point found on many laptops. The middle thumb angle (based on the calibration settings) represents a zero position, while smaller thumb angles represent positive velocities and larger thumb angles represent negative velocities. Therefore, by simply holding the thumb angle steady in some particular offset from the middle angle (with the thumb tip contacting the touch-sensitive surface) the parameter range can be traversed in one of two directions at varying speeds. Figure 5.5 shows the ThumbTrack widget in use, with a visualization that depicts negative velocities with a green arrow pointing to the left, while positive velocities are shown with a green arrow pointing to the right. Note that the size of the arrow varies depending upon the velocity, which ranges from -1 to +1. From an implementation standpoint, it is important to define a "dead zone" around the middle thumb angle since initial testers found it very difficult to precisely set the thumb to the zero velocity position without it. Additionally, since users frequently slide the thumb out from under the index finger to activate the widget, we only enable the parameter control after the thumb enters the dead zone for the first time.

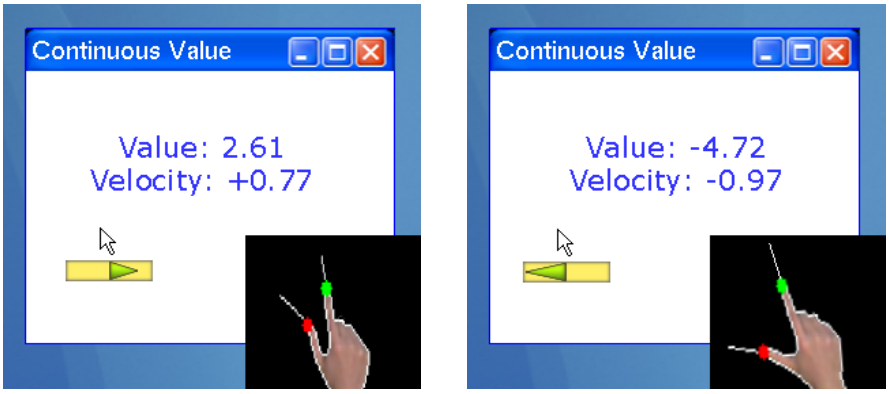


Figure 5.5 - The ThumbTrack widget allows for continuous rate-based parameter control. The direction and size of the green arrow denotes the current velocity.

5.4.3 Discrete Widget Designs

5.4.3.1 ThumbMenu

Figure 5.6 demonstrates dividing the range of thumb angles into *N* discrete zones. The ThumbMenu widget uses such a discretization to allow the thumb to select options from a one-dimensional toolbar as shown in Figure 5.7. This allows for the index finger to perform standard 2D manipulations, while the thumb can make localized command or option selections without requiring a user to change focus towards a system menu. The visualization of the menu/toolbar moves relative to a position defined by the index finger, while the thumb angle changes the active selection (with the thumb making contact with the touch surface). We currently place the thumb menu underneath the cursor, but for direct-touch displays it may be more appropriate to place the thumb menu above the cursor position to reduce the effect of hand occlusions. Menu selections are confirmed using a liftoff approach with the thumb as suggested by Potter et al. [Pott88], but other confirmation techniques such as thumb tapping or pressure are also possible.

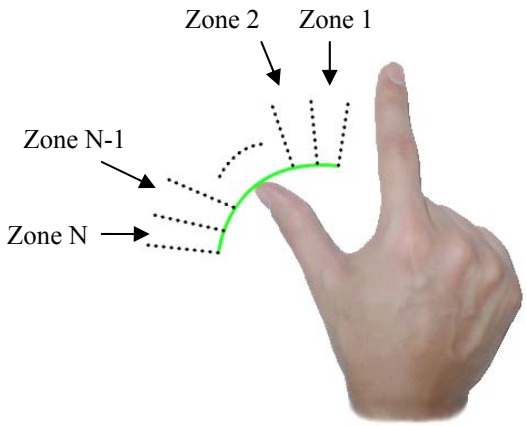


Figure 5.6 - Discretizing the thumb angle into N distinct zones.

5.4.3.2 ThumbToolglass

An alternative to the ThumbMenu is our ThumbToolglass, which uses the same discretization as in Figure 5.6 but with a different visualization and selection technique. The *N* items on the ThumbToolglass are displayed in a circular arrangement around the index finger/cursor position, and the active selection appears directly above this location (right side of Figure 5.7). Therefore, by moving the thumb into the different zones, the entire toolglass rotates so that the corresponding menu item moves to the tip of the index finger or cursor.

The user can then click-through the desired menu item with the index finger in a manner similar to Bier et al.'s original bimanual toolglass [Bier93]. The major advantage of this technique is that it merges command selection and direct manipulation, since the thumb can control the menu item that appears above the index finger, while the index finger can set the focus in 2D and confirm the final item selection simultaneously. Our ThumbToolglass differs from Wu and Balakrishnan's two-fingered toolglass [Wu03] in two significant ways:

- The desired menu item can be highlighted with a one-dimensional motion of the thumb using the ThumbToolglass instead of a two-dimensional thumb motion as in the two-fingered toolglass.
- The two-fingered toolglass requires the user to first position the desired menu item over a target in two-dimensions, followed by a click-through with the index finger. Depending on the location of the menu item in the rectangular toolglass, the user may be required to bend, stretch, or rotate the index finger to complete the selection. Our ThumbToolglass does not require any additional work on the part of the index finger aside from choosing the target location on the canvas and a simple tap or liftoff for the click-through.

Based on these differences, we feel that our ThumbToolglass is a more "ergonomic" single-handed bi-digital toolglass since it considers the various affordances of the thumb and index finger in more detail. A formal user study would be beneficial to determine whether this is actually the case, and if there are any performance differences between the two techniques. Additionally, by using the simple one-dimensional angle of the thumb to make discrete selections, expert users may potentially be able to leverage muscle memory so that toolglass items can be highlighted without the need for visual feedback. This too would require a formal study for confirmation, but with a small *N* the idea seems promising and worthy of future research.

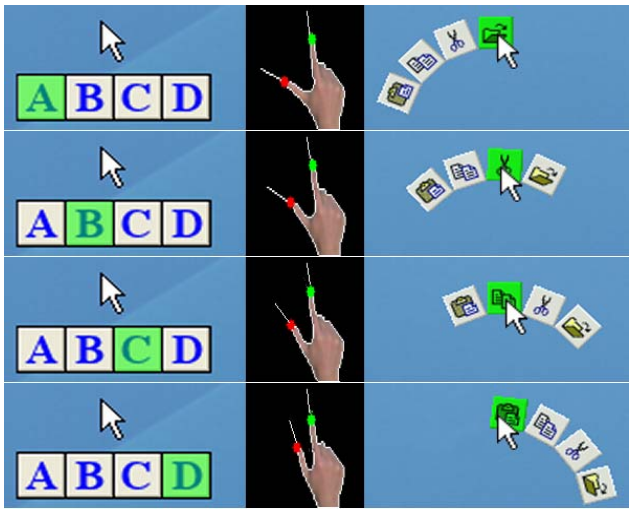


Figure 5.7 - The ThumbMenu (left) and ThumbToolgass (right) allow for locally selecting discrete targets by adjusting the thumb while the index finger controls the position of the cursor to set the focus.

5.4.3.3 ThumbSwitch

Based on the limited precision of existing touch-sensitive hardware as well as the limited range of motion of the thumb, we obviously cannot increase *N* to any arbitrarily large value. As a result, we expect our ThumbMenu and ThumbToolglass will have some upper limit on the number of easily discernible thumb positions for setting discrete parameter values. In situations where we may require traversal of a much larger range of discrete values, we propose the ThumbSwitch widget. This widget divides the thumb range into three equal zones, where the first zone represents an increment zone, the second zone represents a dead zone, and the third zone represents a decrement zone (Figure 5.8a). Therefore, to increment a discrete parameter by one, the user first places the thumb into the dead zone, followed by a quick motion into zone 3 and then back into the dead zone. If the thumb maintains constant contact with the touch surface during this motion, the discrete parameter value is incremented when the thumb returns to the dead zone. A similar motion to and from zone 1 allows the parameter to be decremented. Due to the repetitive flicking motion required to modify values, the ThumbSwitch allows for localized simulation of up/down or left/right arrow keys which are frequently used to change pages in a document or toggle through a linear list of menu items. Figure 5.8b-d shows the visualization of the ThumbSwitch widget, along with the standard cursor that it is attached to. As the thumb is moved, a circular yellow ball underneath the cursor provides continuous feedback about the position of the thumb with

respect to the two active zones. Once the thumb enters one of the zones, the circular ball and corresponding plus or minus icon are highlighted green to denote that a zone was entered successfully. Much like the ThumbMenu, the position of the ThumbSwitch widget visualization can be moved above the cursor to reduce the effect of hand occlusion on direct-touch displays.

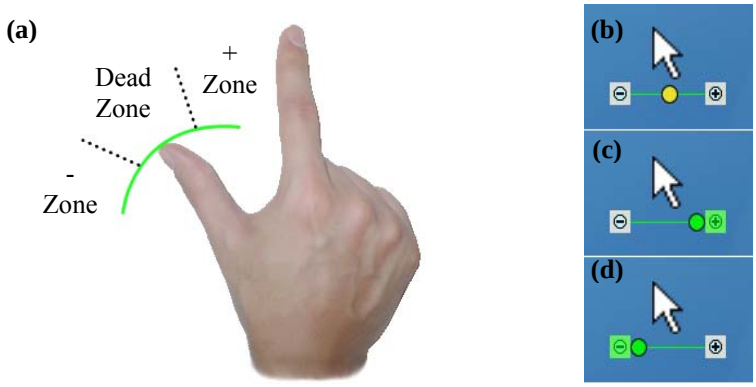


Figure 5.8 - (a) Discrete zones used for the ThumbSwitch widget; (b-d) the on-screen visualization as the thumb is moved into the different zones.

5.5 Initial User Feedback

Six right-handed volunteers with some basic experience in using single-point touch surfaces were given 20 minutes each to explore the functionality of our general-purpose bi-digital widgets. Each user was asked to move the cursor inside of some particular window on the display and then select a particular discrete or continuous value (within some threshold) using the thumb.

In all cases, users felt comfortable with using the thumb as a secondary control, but the comfort level varied depending upon the particular widget. Of the continuous widgets, users felt that the ThumbSlider was the easiest to use, but the ThumbWheel offered the greatest amount of control and precision. The ThumbTrack was generally found to be somewhat more difficult to use than both the ThumbSlider and ThumbWheel, and one user felt it didn't provide the same "instant gratification" that was possible with the other two widgets due to the rate-based control. The poor performance of the resistance-free ThumbTrack is consistent with Zhai's finding which showed that rate control systems are best when used with isometric devices that offer some amount of resistance and are self-centering [Zhai98].

Of the discrete widgets, the ThumbToolglass provided the greatest "wow factor" and was also the most preferred widget, possibly due to its rotational visualization. All users felt that the ThumbSwitch was easy to control, possibly due to the use of only 3 zones, but that traversing a large range of discrete values was time consuming. For both the ThumbMenu and ThumbToolglass, we implemented each with three different *N* values (4, 7, and 10) in order to gauge the effect of adjusting the number of zones. In all cases, we observed that users overshot specific targets more often with 10 zones compared to widgets with only 4 zones. Since selecting discrete targets with the ThumbMenu and ThumbToolglass is essentially a one-dimensional pointing task along a fixed-size axis, Fitts' Law [Fitt54] tells us that increasing the number of zones (which is equivalent to making each zone smaller) will increase the time it takes to select a target. This suggests that there will be some upper limit on *N* in terms of satisfactory user performance, even with infinite touch-sensitive hardware precision.

It is important to note that the visualizations we have chosen for our general-purpose widgets are not always necessary. For example, in certain tasks the current value of the continuous or discrete parameter may be implicit in the interaction, such as when using a ThumbSlider for zooming, or when using a ThumbMenu to change the active tool/cursor in a painting program. Of course, showing a visual depiction of a widget, possibly along with instructions on the valid motions or gestures, may allow new users to become acquainted with the system more quickly [Baud93].

5.6 Experiment

5.6.1 Goals

While the general-purpose widget designs were found to be easy to use in the informal evaluation, what is unclear is how well the asymmetric mapping compares to a symmetric approach. For example, if we generalize Moscovich and Hughes adjustable area cursor [Mosc06] so that the midpoint between the thumb and index finger defines an absolute 2D cursor position while finger span controls a secondary 1D parameter, the widget designs

described earlier could be modified to behave in a symmetric manner. This mapping appears to be natural since it mimics how real-world objects are often grasped, where the midpoint between the thumb and index finger is placed close to the centre of mass of an object to maximize grasp stability [Lede03].

In a similar manner, it is not clear whether reversing the roles of the fingers in the asymmetric mapping would have any effect on performance. Arguably, using the index finger for the primary 2D control seems intuitive since this is typically the finger which performs standard manipulations on touch-sensitive surfaces. However, our widget designs could easily use an alternative mapping where the thumb defines the 2D cursor position and the relative position of the index finger could be used to adjust finger span to control the secondary 1D parameter.

The goal of this experiment is to therefore determine whether there are any performance differences when the roles of the thumb and index finger are reversed in an asymmetric task, as well as how these two asymmetric mappings compare to a symmetric mapping during a compound selection task. To denote the different mappings we will use the following terms:

- The *Index Finger Cursor* will refer to the asymmetric mapping where the index finger performs the primary 2D operations and the relative position of the thumb performs the secondary 1D control.
- The *Midpoint Cursor* will refer to the symmetric mapping where the average position between the thumb and index finger defines the 2D position while the distance between the two fingers controls the secondary 1D parameter.
- The *Thumb Cursor* will refer to the asymmetric mapping where the thumb performs the primary 2D operations and the relative position of the index finger performs the secondary 1D control.

5.6.2 Apparatus

An upright 24” LCD display running at a resolution of 1024x768 pixels was used to present visual stimuli. The experimental software was run on a P4 3.0GHz PC running Windows XP.

Thumb and index finger positions of a user's right hand were detected by placing a DiamondTouch system [Diet01] horizontally on a desk between the user and the display (Figure 5.9a). We used the DiamondTouch instead of the Visual Touchpad since it provides significantly higher positional accuracy which is desirable for our formal study. However, since the DiamondTouch only provides a bounding box around two or more contact positions on its surface, it is very difficult to disambiguate between fingertips. Nevertheless, the DiamondTouch is capable of detecting single contact points from up to 4 different users without any ambiguity by requiring users to sit on special receiver pads that are connected to different input ports on the DiamondTouch device. Therefore, to overcome the finger disambiguation problem, we outfitted a glove with conductive pads at the thumb and index finger tip positions and connected each of the fingers (via RCA cable) to the DiamondTouch so that each finger was recognized as a unique user (Figure 5.9b).

Contact information on the DiamondTouch is reported at approximately 22Hz with an interpolated sensor resolution of 2752x2064 and a physical diagonal touch-surface measurement of 107cm, which allows for detecting fingertips with an accuracy of up to 0.03cm. The corners of the touch surface were mapped to the corners of the 1024x768 display, which provided contact information with sub-pixel accuracy.

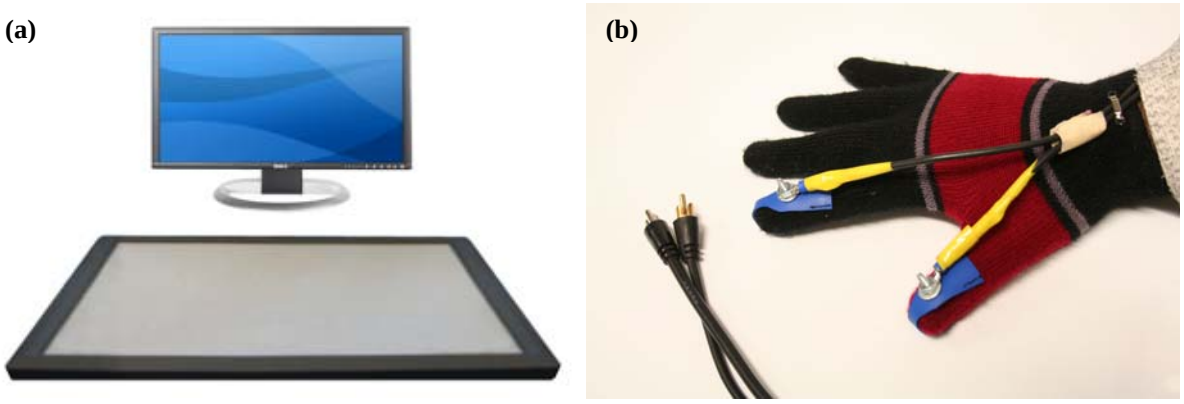


Figure 5.9 - (a) Experimental configuration with the DiamondTouch and LCD display; (b) A simple glove outfitted with conductive pads at the thumb and index finger positions to facilitate finger disambiguation.

5.6.3 Participants

Nine participants, three female and six male, 20-33 years old, volunteered for the experiment. All were right-handed and had little to no experience with multi-point touch-sensitive devices.

5.6.4 Task and Stimuli

A compound target selection task was used, where the user was required to position a standard 2D cursor into a red square target while finger span was used to adjust the radius of a black circular ring around the square target so that it matched the radius of a red target ring. Figure 5.10 shows an example of this compound 2D+1D target, where the red square represents the primary 2D target and the red ring represents the secondary 1D target.

Note that the black ring is effectively a visual representation of the secondary 1D control, and its center is located at the center of the red square. In a pilot study we originally attached the center of the black ring to the 2D cursor position so that the ring would always move with the cursor. However, this resulted in users visually coupling the 2D and 1D tasks into a single ring docking task. In other words, users would ignore the 2D cursor position and instead only focus on the black ring for both ring size matching and position alignment. In real world scenarios, our widgets are designed for tasks where the 2D position defines the primary focus of a manipulation while the 1D task is used to support the 2D task. For example, a drawing tool where the 2D control defines the position of a drawing tip on a canvas and the 1D control adjusts the colour of the drawing tip does not allow for visually coupling the two tasks. Therefore, in the experiment we decided to center the black ring on the target square instead of moving it with the cursor so that the experimental task resembled the style in which the widgets will be used in actual user interfaces.

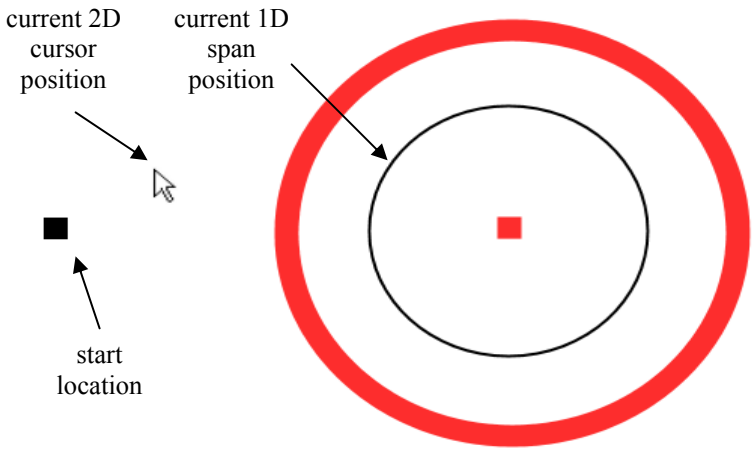


Figure 5.10 - An example of a compound target consisting of a red ring with a red square at the center. The arrow cursor represents the current 2D position, while the black ring represents the current 1D position based on finger span. All trials start with the 2D cursor inside of a black square located at the center of the screen with finger span less than 1cm.

Each trial began by asking the user to move the 2D cursor into a small black square located at the center of the screen while setting the span between the thumb and index finger to less than 1cm. Subsequently, a compound target would randomly appear either to the left or right of the center start position. The amplitude to the 2D target and the size of the red square was chosen randomly from a predetermined set of possible values. Similarly, the amplitude (radius) of the red target ring and its width (thickness) were also chosen randomly from a predetermined set of values.

When the cursor was inside of the square target, the color of the square would change to green. Similarly, when the black ring was within the thicker red ring, the thicker ring would change to green. To confirm the selection and complete the trial, the user was required to hold the fingers steady for one full second while both the target square and target ring were green. We used the three different cursor control conditions described earlier: *Index Finger Cursor*, *Midpoint Cursor*, and *Thumb Cursor* (Figure 5.11). For secondary amplitudes that were larger than a user's maximum finger span, users were able to re-clutch (similar to the *ThumbWheel*) in order to adjust the black ring to the desired size.

The 2D target positions were chosen to eliminate large finger spans from reaching the edges of the touch-sensitive surface. We also decided to randomly select the left/right directions of the targets from the starting position instead of using a reciprocal left/right target direction in order to reduce the chance of a user anticipating where the next target would appear. The decision to use a ring to visually represent the secondary 1D task was based on keeping a consistent stimulus-response for finger span across the different cursor control mappings. If we had chosen a horizontal or vertical slider, for example, the perception of what a large span or small span corresponded to in terms of a 1D position would have changed across the different mappings. For example, with the Index Finger Cursor and a vertical slider, it seems intuitive that minimum span should be visually mapped to the top of the slider while maximum span should denote the bottom of the slider (based on the direction of motion of the thumb). However, with a Thumb Cursor and a vertical slider, it seems more intuitive that the minimum span should map to the bottom of the slider and maximum span should visually map to the top of the slider (based on the direction of the index finger's motion). The ring, however, is consistent across the different mappings since its radius expands equally in all directions from the 2D centre point based on finger span.

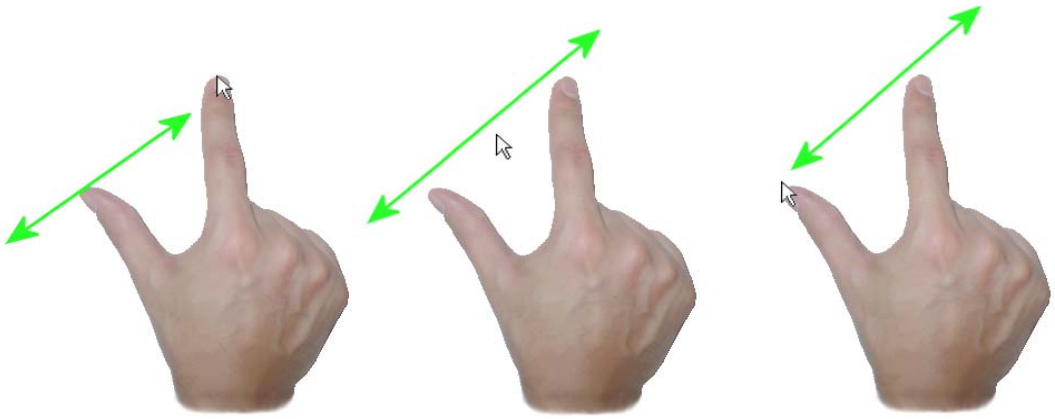


Figure 5.11 - The three cursor control mappings: (left) Index Finger Cursor; (center) Midpoint Cursor; (right) Thumb Cursor.

5.6.5 Procedure and Design

We used a within-participants full factorial design with repeated measures. Independent variables were cursor condition (*Index Finger Cursor*, *Midpoint Cursor, Thumb Cursor*), the distance/amplitude to the 2D target (A = 75, 150), the width of the 2D target (W = 4, 8), the distance/amplitude to the secondary target (SA = 75, 150), and the width of the secondary

target (*SW* = 4, 8). All units are measured in millimeters with respect to the touch-sensitive surface. The particular values of the widths and amplitudes were chosen to test different combinations of the Index of Difficulty (ID) [Fitt54] for the primary and secondary tasks.

Participants were randomly assigned to 3 groups of 3 participants each. Within each group, participants were exposed to all three cursor conditions, with the order of appearance balanced using a Latin square. For each cursor condition, participants completed a session of 4 blocks, where each block consisted of trials for all 16 *A*-*W*-*SA*-*SW* conditions, repeated 4 times in random order. In summary, the experiment design consisted of:

9 participants x
3 cursor control mappings x
4 blocks x
2 primary target amplitudes (*A* = 75, 150) x
2 primary target widths (*W* = 4, 8) x
2 secondary target amplitudes (*SA* = 75, 150) x
2 secondary target widths (*SW* = 4, 8) x
4 repetitions
= 6912 total selection trials

Participants were also given a warm-up block at the start of each cursor condition in order to become familiar with the task. Participants were informed that they could take breaks between individual trials as well as between each cursor control condition. Participants were also instructed to complete each trial as quickly and as accurately as possible. Finally, participants were instructed to keep both the thumb and index finger on the touch-sensitive surface during each trial so that finger kinematics could be measured continuously. In total, the experiment lasted approximately 1.5 hours for each participant.

5.6.6 Dependent Variables

Dependent variables were movement time (*MT*), which is defined as the time it takes from the start of a trial to when the cursor is inside of the square target and the black ring is within the thicker target ring for one second; simultaneity of control (*SOC*), proposed by Masliah

and Milgram [Masl00], which represents the percentage of time during a trial that the error for the primary and secondary tasks was being reduced in parallel; efficiency (*EFF*), also proposed by Masliah and Milgram, which is the ratio between the optimal trajectory length and actual trajectory length for the primary and secondary tasks; and clutching (*C*), which measures the number of times in a trial that clutching was necessary to complete the secondary task.

5.6.7 Hypotheses

Based on the work by Raj and Marquis [Raj99], which suggests that the index finger is the preferred digit in simple selection tasks, we expect that:

H1. The *Index Finger Cursor* will outperform the *Thumb Cursor* in terms of trial completion time.

H2. The *Index Finger Cursor* will exhibit higher simultaneity of control than the *Thumb Cursor*.

H3. The *Index Finger Cursor* will outperform the *Thumb Cursor* in terms of movement trajectory efficiency.

H4. The *Index Finger Cursor* will result in less clutching operations than the *Thumb Cursor*.

Similarly, since the *Thumb Cursor* is arguably somewhat awkward due to its reliance on the thumb for defining the 2D focus of a manipulation, we feel that:

H5. The *Midpoint Cursor* will outperform the *Thumb Cursor* in terms of trial completion time.

H6. The *Midpoint Cursor* will outperform the *Thumb Cursor* in terms of movement trajectory efficiency.

H7. The *Midpoint Cursor* will result in less clutching operations than the *Thumb Cursor*.

The degrees of freedom for the *Midpoint Cursor* are more integrated than both the *Thumb Cursor* and the *Index Finger Cursor*. Due to this higher integration, we hypothesize that:

H8. The *Midpoint Cursor* will exhibit higher simultaneity of control than both the *Thumb Cursor* and *Index Finger Cursor*.

However, this high integration of the *Midpoint Cursor* also implies that finger span adjustments have the potential to inadvertently modify the position of the midpoint if the two fingers are not moved in a perfectly symmetric manner. Therefore, we hypothesize that:

H9. The *Index Finger Cursor* will outperform the *Midpoint Cursor* in terms of trial completion time when selecting small 2D targets.

H10. The *Index Finger Cursor* will outperform the *Midpoint Cursor* in terms of movement trajectory efficiency when selecting small 2D targets.

5.6.8 Results

We removed outliers from the set of data, where a trial was considered an outlier if the *MT* was beyond 2 standard deviations from the mean task completion time. A total of 314 trials were removed, representing 4.5% of the data. Outliers occurred when, after repeated use, the conductive tips of the glove rotated slightly out of place, reducing the amount of contact between the fingertips and the touch-sensitive surface. This was temporary, however, since the tips were easily rotated back into place for subsequent trials.

5.6.8.1 Trial Completion Time

One of the primary goals of the experiment was to determine if there were any differences between the three cursor control mappings in terms of trial completion time. Analysis of variance showed a significant effect of cursor control mapping on *MT* ($F_{2,16}$=11.87, p<0.01), with mean movement times of 3.87s, 3.85s, and 4.42s for *Index Finger Cursor*, *Midpoint Cursor*, and *Thumb Cursor* respectively. Pairwise means comparison showed significant difference between *Index Finger Cursor* and *Thumb Cursor* (p<0.01), and *Midpoint Cursor* and *Thumb Cursor* (p<0.01). However, there was no significant difference between *Index Finger Cursor* and *Midpoint Cursor*. Therefore, hypotheses H1 and H5 were confirmed.

We also wanted to determine if there was any difference in movement time across blocks for each of the different control mappings. Repeated measures ANOVA showed a significant effect of block on *MT* ($F_{3,24}$=6.84, p<0.01), with a pairwise means comparison showing a significant difference between the first block and all other blocks (p<0.05), but there was no significant difference between any other block combinations. In all cases, mean completion times went down as the block number increased. There was also no significant cursor

mapping x block interaction on *MT* which suggests that the learning effect was consistent across the three cursor mapping conditions. Figure 5.12 shows the average *MT* for the three cursor control mappings across the four blocks.

We found a significant cursor mapping x *SW* interaction on *MT* ($F_{2,16}$=4.57, p<0.05). The *Thumb Cursor* performed significantly worse than both the *Index Finger Cursor* and *Midpoint Cursor* for both *SW*=4 and *SW*=8 (p<0.01), but the *Index Finger Cursor* and *Midpoint Cursor* did not have any significant differences across the two *SW* conditions. There was no significant cursor mapping x *W* interaction on *MT*, however, so hypothesis H9 was not confirmed.

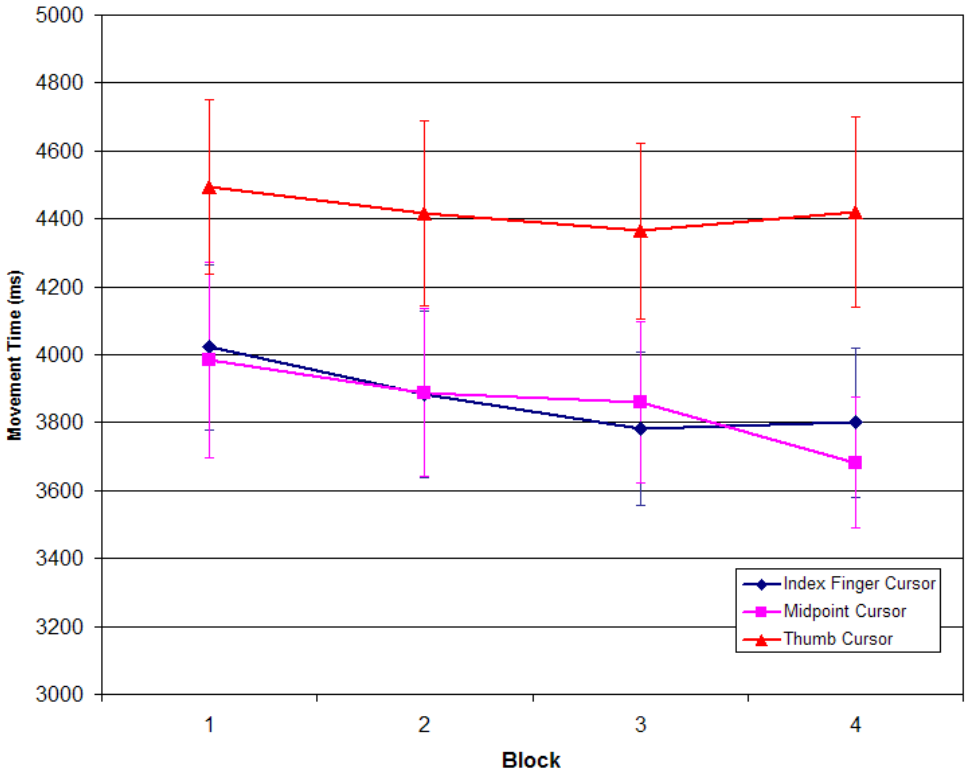


Figure 5.12 - Average movement times by block for each cursor control mapping (with standard error bars).

5.6.8.2 Simultaneity of Control

The *SOC* gives us an estimate of the percentage of time during a trial in which parallel movement occurred. An important aspect of this metric is that only parallel movements which reduce the error (between the current and goal position) for the primary and secondary degrees of freedom are taken into account. Overall, average *SOC* values were 47%, 59%, and

42% for the *Index Finger Cursor*, *Midpoint Cursor*, and *Thumb Cursor* respectively. A repeated measures ANOVA showed a significant effect of cursor mapping on *SOC* ($F_{2,16}$=10.82, p<0.01). Pairwise means comparison showed a significant difference in *SOC* between the *Index Finger Cursor* and the *Midpoint Cursor* (p<0.05), and a significant difference between the *Thumb Cursor* and *Midpoint Cursor* (p<0.01). However, there was no significant difference between the *Index Finger Cursor* and *Thumb Cursor*. Therefore, hypothesis H8 was confirmed, but hypothesis H2 was not.

There was also a significant effect of block number on *SOC* ($F_{3,24}$=5.30, p<0.01), with a pairwise means comparison showing a significant difference only between the first block and the other three blocks (p<0.05). The average overall *SOCs* for blocks 1 to 4 were 47.8%, 49.8%, 49.3%, and 50.3%. There was no cursor mapping x block interaction, however, which suggests that the slight improvement in *SOC* across blocks was consistent for the various cursor mappings. Figure 5.13 shows the average *SOC* for the three cursor control mappings across the four blocks.

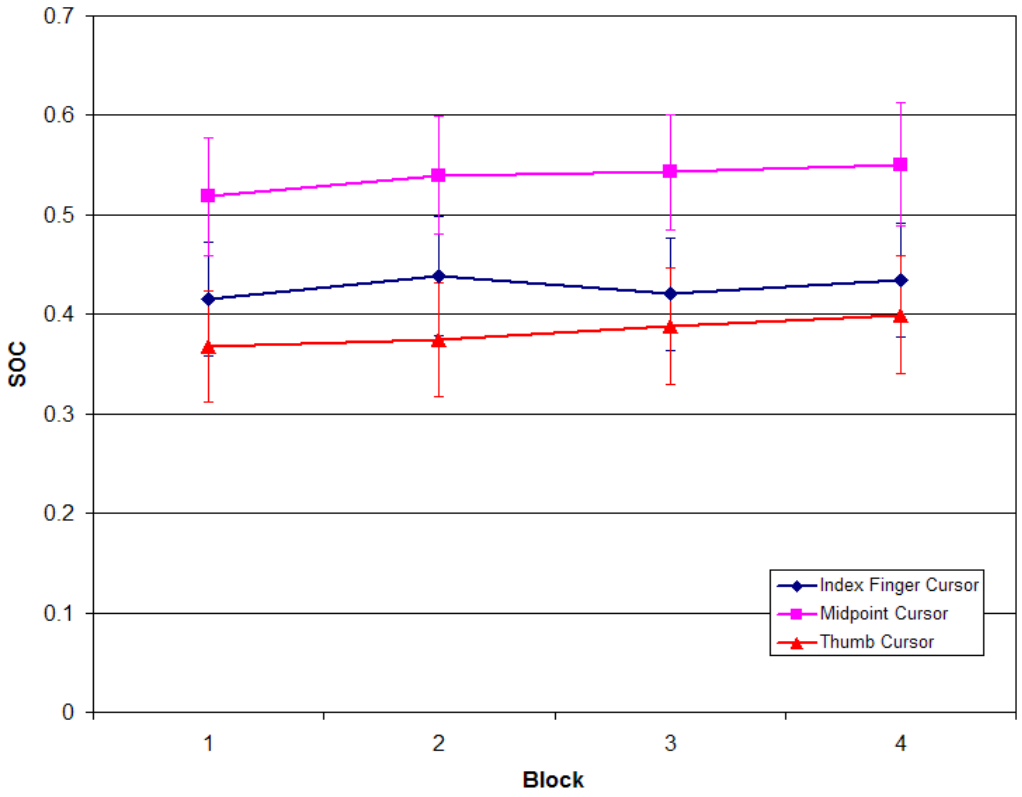


Figure 5.13 - Average SOC by block for each cursor control mapping (with standard error bars).

5.6.8.3 Efficiency

The *EFF* metric provides an estimate of how efficient the trajectory was for the primary and secondary selection tasks, with a range of between 0 and 1, where 1 is the most optimal trajectory. We used the following equation to compute *EFF*, based on the original formulation by Masliah and Milgram [Masl00]:

$$EFF = \frac{1}{2}\frac{OPT_A}{ACT_A} + \frac{1}{2}\frac{OPT_B}{ACT_B}$$

where OPT_i represents the optimal length of the trajectory for the *i*-th degree of freedom, and ACT_i represents the total actual error reduced for the *i*-th degree of freedom. We denote *A* to represent the 2D task, and *B* to represent the 1D task. The ACT_i component is computed by accumulating the instantaneous error reduction $-dE_i(t)/dt$ at time *t* for the degree of freedom *i*, where E_i = goal position – cursor position. For $dE_i(t)/dt >= 0$, which occurs when the error may increase, the instantaneous error reduction is set to zero.

One theoretical limitation with this metric is the potential for two apparently different paths, from a human efficiency perspective, to have the same *EFF* value (see Figure 5.14). For this reason, it is important to not consider this metric in isolation, but rather in conjunction with other metrics such as task completion time. However, from a practical standpoint, we expect that users will normally choose a path that somewhat resembles the most direct one between two points, so we feel that this metric is sufficient for our purposes.

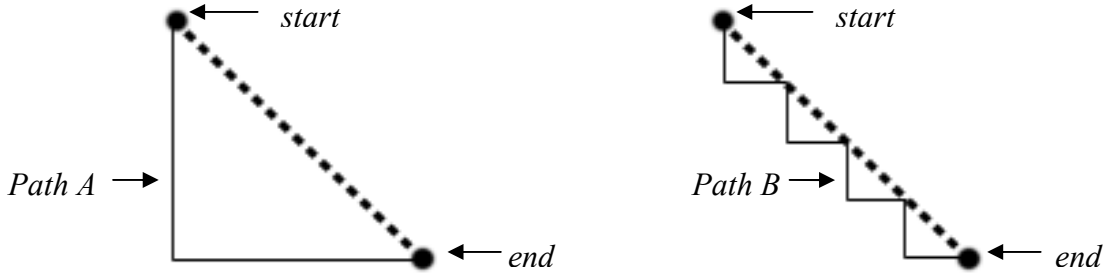


Figure 5.14 – Two different paths with the same *EFF* value. Path A, however, is clearly more efficient in terms of human performance compared to Path B.

Analysis of variance showed no significant difference in *EFF* for the different cursor mappings. On average, the *EFF* values were 0.76, 0.69, and 0.73 for the *Index Finger*

Cursor, *Midpoint Cursor*, and *Thumb Cursor* respectively. Therefore, hypotheses H3 and H6 were not confirmed. There was also no significant difference in *EFF* values across blocks.

We found a significant cursor mapping x *A* interaction on *EFF* ($F_{2,16}$=5.09, p<0.05). A pairwise means comparison found a significant difference between *EFF* for the *Index Finger Cursor* and *Midpoint Cursor* with *A*=75 (p<0.05), and also between the *Index Finger Cursor* and *Midpoint Cursor* with *A*=150 (p<0.05). With *A*=75, the mean *EFF* for the *Index Finger Cursor* and *Midpoint Cursor* were 0.73 and 0.67 respectively. Similarly, with *A*=150, the mean *EFF* for the *Index Finger Cursor* was 0.78 and for the *Midpoint Cursor* the mean *EFF* was 0.71.

We also found a significant cursor mapping x *SA* interaction on *EFF* ($F_{2,16}$=20.82, p<0.01). Pairwise means comparison showed a significant difference in *EFF* between the *Index Finger Cursor* and the *Midpoint Cursor* (p<0.01), and between the *Midpoint Cursor* and the *Thumb Cursor* (p<0.01), both with *SA*=150. The mean *EFF* with *SA*=150 was 0.72 for the *Index Finger Cursor*, 0.59 for the *Midpoint Cursor*, and 0.68 for the *Thumb Cursor*.

There was no cursor mapping x *W* interaction on *SOC*, so hypothesis H10 was not confirmed.

5.6.8.4 Clutching

The number of clutching operations (*C*) gives an estimate of how much effort is required to perform a secondary selection. A repeated measure ANOVA showed no significant effect of cursor control mapping on *C*. There was also no effect of block number on *C*, nor was there a significant cursor control x block interaction. Additionally, there were no significant interactions between cursor control and the various combinations of *A*, *W*, *SA*, and *SW* in terms of *C*. Therefore, hypotheses H4 and H7 were not confirmed. Overall, the average number of clutching operations was 2.29, 1.70, and 2.32 for the *Index Finger Cursor*, *Midpoint Cursor*, and *Thumb Cursor* respectively.

5.6.8.5 Subjective Rating

A post experiment questionnaire was used to collect subjective ratings. For each cursor control mapping, participants were asked to rate their perceived task completion speed on a

scale of 1 to 7, where 1 was "very slow" and 7 was "very fast". A repeated-measures ANOVA determined that means for the subjective speed rating differed significantly across cursor control mappings ($F_{2,16}$=63.36, p<0.01). Pairwise means comparison showed a significant difference between the *Index Finger Cursor* and *Thumb Cursor* (p<0.01), and the *Midpoint Cursor* and *Thumb Cursor* (p<0.01), but no significant difference between the *Index Finger Cursor* and *Midpoint Cursor*. The average ratings for the *Index Finger Cursor*, *Midpoint Cursor*, and *Thumb Cursor* were 5.78, 5.83, and 2.94 respectively.

Participants were also asked to rate their perceived accuracy with each cursor control mapping, where 1 was "very inaccurate" and 7 was "very accurate". A repeated-measures ANOVA determined that means for the subjective accuracy rating differed significantly across cursor control mappings ($F_{2,16}$=26.57, p<0.01). Pairwise means comparison showed a significant difference between the *Index Finger Cursor* and *Thumb Cursor* (p<0.01), and the *Midpoint Cursor* and *Thumb Cursor* (p<0.01), but no significant difference between the *Index Finger Cursor* and *Midpoint Cursor*. Average results were 5.83, 5.67, and 2.78 for the *Index Finger Cursor*, *Midpoint Cursor*, and *Thumb Cursor* respectively.

Finally participants were asked to rate their overall comfort for each cursor control mapping, where 1 was "very uncomfortable" and 7 was "very comfortable". Similar to the speed and accuracy subjective rankings, a repeated-measures ANOVA determined that means for the subjective comfort rating differed significantly across cursor control mappings ($F_{2,16}$=24.19, p<0.01). Pairwise means comparison showed a significant difference between the *Index Finger Cursor* and *Thumb Cursor* (p<0.01), and the *Midpoint Cursor* and *Thumb Cursor* (p<0.01), but no significant difference between the *Index Finger Cursor* and *Midpoint Cursor*. Average results for comfort were 5.78, 6.0, and 3.06 for the *Index Finger Cursor*, *Midpoint Cursor*, and *Thumb Cursor* respectively.

5.6.9 Discussion

Overall, the results of the experiment and questionnaire show that the *Index Finger Cursor* and *Midpoint Cursor* perform similarly in terms of the task completion time, which validates

our asymmetric interaction style as an alternative to many of the existing symmetric bi-digital interaction techniques.

The *Midpoint Cursor* exhibited significantly higher parallelism than the *Index Finger Cursor*, but this did not equate to faster task completion times. One surprising finding was the higher efficiency that was exhibited by the *Index Finger Cursor* and *Thumb Cursor* in comparison to the *Midpoint Cursor* for both of the primary target amplitudes and the larger secondary amplitude. This suggests that span adjustments with the *Midpoint Cursor* increased the distance traveled by the 2D cursor throughout each trial, which is related to our motivations behind hypotheses H9 and H10.

Not surprisingly, the *Thumb Cursor* performed significantly worse than both the *Index Finger Cursor* and *Midpoint Cursor* in terms of task completion time, which suggests that the mapping of the primary and secondary tasks for the *Thumb Cursor* was less natural compared to the other two mappings.

Since the finger mappings allow for performing compound selections, a possible explanation for the experimental results could be attributed to the idea of a kinematic chain for bi-digital input in a manner similar to Guiard's model for bimanual asymmetry [Guia87]. Unlike Guiard's work, however, the concept of a kinematic chain can be applied more readily to the fingers of a single hand based on the hand's actual hierarchic structure. In other words, for the case of the *Midpoint Cursor*, the midpoint between the thumb and index finger can effectively be thought of as a proxy for the position of the hand. The two fingertips then act as distal elements in a kinematic chain that organize their symmetric movements relative to the position of the proximal hand.

While the *Index Finger Cursor* defines manipulations in an asymmetric manner, the mapping in terms of a kinematic chain is essentially the same: the tip of the index finger acts as a proxy for the position of the proximal hand, while the tip of the thumb acts as the distal element whose movements occur relative to the position of the hand. Such a conceptual model seems plausible since it is very easy to move the thumb in opposition to a stationary

index finger without affecting the rest of the hand. The opposite is not true, however: performing adduction-abduction movements of the index finger towards the thumb while the thumb remains stationary is very difficult when attempting to keep the rest of the hand stable. Therefore, in terms of a kinematic chain, the thumb is a poor proxy for the hand position when the index finger moves relative to it. As a result, it is not surprising that the *Thumb Cursor* performed significantly worse than the other two bi-digital mappings, while the *Index Finger* and *Midpoint Cursor* performed similarly.

Clearly, from a design perspective, the results of the experiment suggest that how roles are assigned to the fingers in a bi-digital task is an important consideration. What remains unclear, however, is whether a designer should choose the *Index Finger Cursor* or the *Midpoint Cursor* when designing task-specific bi-digit widgets. From a multi-point input device perspective, the *Midpoint Cursor* is appealing since determining the midpoint between the thumb and index finger does not require any finger disambiguation algorithm. However, the *Midpoint Cursor* has a disadvantage for tasks which require frequent switching between one and two-fingered operations, since the 2D cursor position will suddenly jump from the index finger position to the midpoint between the thumb and index finger during a transition, resulting in a sudden focus change that will require a user to retarget the cursor back to its original location. The *Index Finger Cursor* does not have this problem, so transitioning between single-finger and two-finger tasks is much more fluid. Therefore, when designing practical, task-specific bi-digit widgets, we suggest the following simple guidelines:

- For tasks that require a user to perform bi-digital manipulations continuously for extended periods, the symmetric *Midpoint Cursor* is more appropriate.
- For tasks that require frequent switching between single-finger and bi-digital states, or for single-finger tasks that only require occasional bi-digit widget usage, the *Index Finger Cursor* is more appropriate.

For touch-sensitive devices that do not automatically disambiguate fingers but wish to use bi-digit widgets based on the *Index Finger Cursor*, a variety of effective disambiguation approaches have been proposed such as simple fingertip tracking algorithms [Reki02] and finger placement orders/protocols [Wu03, Benk06].

5.7 Practical Widget Designs

The designs described previously are general purpose widgets that can be customized for various localized tasks on multi-point touch-sensitive surfaces in order to merge command selection or parameter adjustments with direct manipulation. In this section we demonstrate two practical instantiations of the widgets using the Visual Touchpad: a continuous ThumbSlider widget that is used to adjust the zoom level of a small region around the cursor, and a discrete ThumbToolglass that allows for localized selections of common system commands and operations.

The ThumbToolglass is configured with the following four options: open, cut, copy, and paste (Figure 5.1). These options correspond to commands that are frequently issued in many standard graphical user interfaces, but which must typically be activated on touch-screens by accessing system menus or in some cases by simulating a right mouse button click and then selecting from a popup menu. Our ThumbToolglass allows these commands to be issued more fluidly by simply selecting the appropriate toolbar item with the thumb and then single-tapping on a target object with the index finger.

One common issue with many touch-screens is the difficulty in selecting small targets due to both hardware precision limitations as well as the relatively large size of the tip of the index finger. To remedy this problem, we use the ThumbSlider to locally adjust the zoom level of a small rectangular region centered on the position of the index finger (Figure 5.15). Therefore, with the index finger in the tracking state, zooming can be modified by making the appropriate motion with the thumb. This contrasts with the bimanual precision selection techniques proposed by Benko et al. [Benk06], where the non-dominant hand was used to adjust the zoom level instead of using two fingers from the same hand. By zooming, the mapping of the touchpad locally changes so that subsequent manipulations with the single index finger occur at the new zoom level, which allows for more accurate selections. We currently clamp the ThumbSlider's range between 1 and 3, which corresponds to a zoom factor. When in a zoomed state, index finger manipulations that occur outside of the zoom region cause the zoom level to return back to 1.

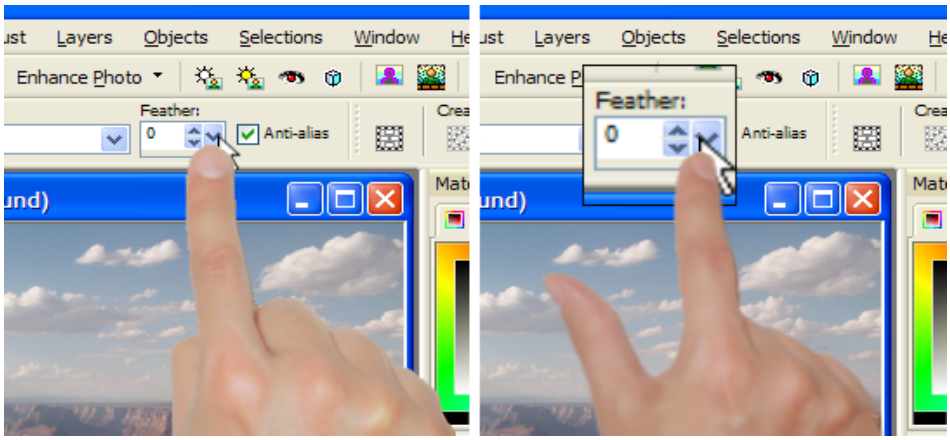


Figure 5.15 - The ThumbSlider is being used to increase the zoom in a small rectangular region around the cursor, allowing for more precise selections.

5.8 Summary

Drawing on concepts from the bimanual interface literature, we presented a taxonomy of bi-digital-dependent tasks that leveraged the independent positioning capabilities of the thumb and index finger of a single hand on multi-point touch-sensitive surfaces. Such tasks may not only be used to complement the functionality of existing bimanual interfaces, but they may also be beneficial in situations where bimanual interactions are difficult to perform, such as when using portable hand-held touch-sensitive devices. We explored the design space of such asymmetric bi-digital tasks by presenting a variety of general-purpose bi-digit widget designs which allowed the thumb to act as a localized secondary control to modify parameters for the index finger's primary operations. We also presented the results of a formal experiment that validate this asymmetric finger mapping. We feel that these explorations and guidelines can significantly influence future multi-finger user interface designs since there is currently no standard interaction paradigm for multi-point touch-sensitive surfaces.

While the initial investigations presented in this chapter are encouraging, we feel that we have only just scratched the surface of the rich design space of single-handed asymmetric bi-digital tasks. In addition to the various avenues of research described earlier, it would be interesting to evaluate the effect of long-term usage of our widgets. For example, guitar and piano players frequently perform various hand exercises in order to increase the amount of

independent control in their fingers [Sue02]. Similarly, studies from the neuroscience literature show that blind Braille readers have an expanded cortical representation of their fingers and a higher spatial acuity with their fingertips than sighted individuals [Bove00]. Both of these results suggest the amount of independent thumb and index finger control when using our widgets may improve over time. Alternatively, if finger independence can be improved with repetitive exercise, then it may be the case that other fingers could potentially be trained and used for various secondary or even tertiary manipulations. However, work by Santello et al. shows that over 80% of the variance in hand postures during everyday grasping tasks can be accounted for by the first two principal components, which suggests that hand posture primarily involves the coordination of two synergies [Sant98]. Therefore, it may be difficult to gain significant extra control from the remaining three fingers. Additionally, the ring and little fingers are attached to the same set of flexor and extensor muscles in the forearm, which suggests a physiological limit to the amount of independence that they can develop regardless of the amount of exercise or repetition [Sue02].

Another possibility for future work is to investigate bi-digital widget designs that leverage the affordances of other degrees of freedom such as hover or pressure [Ramo04]. Additionally, it would be beneficial to design more task-specific widgets in the various domains where multi-point touch surfaces may be utilized, such as for large display interactions or multi-user interactive tabletops.

It would also be worthwhile to evaluate the performance of our widgets with different surface arrangements. Our current prototype with the Visual Touchpad allows interactions to be performed on a horizontal touch-sensitive surface while the image of the hand or fingertip positions are projected onto an upright display. Although our widget designs are general enough to be used directly with upright multi-touch-sensitive displays such as the SMARTBoard [Smar05], we should examine how the change in the posture of the wrist joint affects the range of motion of the thumb.

Chapter 6

An Evaluation of Finger Span Perception for Bi-digital Input

6.1 Introduction

As discussed in Chapter 5, the thumb and index finger have been shown to offer the largest amount of independent control [Hage00]. Therefore, it is not surprising that many existing multi-finger interaction techniques leverage these two fingers for single-handed operations, where the span between the thumb and index finger is used to manipulate a continuous parameter for operations such as zooming or resizing objects. To the best of our knowledge, however, there has not been any formal study of the capabilities and limitations of finger span from a user interface perspective.

In this chapter, we present a controlled experiment that tries to answer a number of important questions that arise when the thumb and index finger are used to control a discrete bi-digit widget, such as: how many discrete targets is a user capable of easily discriminating between with the thumb; can an expert user perform the secondary discrete selections with little or no visual feedback; and do users naturally adopt a serial or parallel strategy when required to select a discrete secondary target as well as directly manipulate the position of the cursor? Based on the experimental results, we present three advanced bi-digit widget designs for finger span that allow for simultaneous direct manipulation and command selection using a

single hand, with two designs demonstrating smooth transitioning from novice to expert usage.

6.2 Related Work

Clearly, the effectiveness of each bi-digit widget design from Chapter 5 relies on how well humans can perceive and control different finger spans: the better the perception and control, the higher the accuracy of the parameter adjustment widgets. While the HCI literature has not addressed such human performance issues related to finger span, the motor control, perception, and psychophysics literature provide a rich set of results that user interface designers might build upon. Jastrow [Jast86] asked participants to match the span between the thumb and index finger to the length of a viewed line, without visibility of the hand. Results showed that participants consistently overestimated line length based on finger span. A second experiment measured the inverse, where participants matched the width of blocks held between the thumb and index finger to a length on ruled paper, with results showing that the chosen line lengths were consistently shorter than actual span. Taken together, these results indicate that perceived finger span is less than lines of equal length.

van Doren [Vand95] performed a similar set of experiments to determine whether tactile information affected finger span perception. Results showed that when participants matched line lengths to finger span while pinching blocks of various sizes, the matching was well approximated by an accelerating power function. However, when there were no tactile cues, the matching function between perceived finger span and line length was proportional. In both cases, however, the results suggest that perceived span is roughly 90% of actual span on average, which is consistent with Jastrow's findings. van Doren's results also showed that there was no significant difference in terms of finger span perception accuracy between the left and right hands.

Santello and Soechting [Sant97] also looked at how accurately people could control finger span between the thumb and another finger. Their first experiment required participants to estimate the size of an object presented visually, without visibility of their right hand, using finger span. Their second experiment also required participants to estimate the size of an

object with finger span, but the object was sensed haptically with their left hand instead of visually. In both cases, the results suggested that people were able to accurately estimate object size, with the small errors tending to be negative (i.e. finger span < object size), and with errors increasing as the object size increased. This result, however, conflicts with the previous studies by Jastrow and van Doren, but it is not clear as to why this is the case. A potential explanation could be attributed to the different stimuli that were used, since van Doren and Jastrow used line length estimation while Santello and Soechting estimated the size of objects such as cubes and cylinders. Nevertheless, the differences are small, with both results showing that participants can adjust finger span with high accuracy in response to visual or haptic stimuli. Santello and Soechting also found that participants were equally accurate when using the index, middle, and ring fingers in opposition with the thumb, but the little finger and thumb combination resulted in a slight decrease in performance.

Finger span has also been used by flavour chemists for categorizing the intensity of odours [Ekma67, Etie99], and experimental data shows that the space of finger spans that participants generate to classify intensities is highly correlated with the space of theoretical intensities. From the perspective of user interfaces, this is encouraging since it suggests that users may be able to remember particular spans for selecting various commands.

In summary, the motor control and psychophysics literature tells us that humans are quite proficient at estimating finger span when presented with visual or haptic stimuli, but there are conflicting results as to how finger span is perceived. Meanwhile, the HCI literature has demonstrated a variety of multi-finger techniques and widgets that use finger span for controlling continuous parameters, but there has not been any systematic investigation into how finger span performs in a user interface, particularly when span is interpreted as a discrete parameter for selecting commands or states. Therefore, we feel that this is a fruitful direction for further research.

6.3 Experiment

6.3.1 Goals

The objective of this experiment is to investigate human ability when using the span between the thumb and index finger to control a discrete secondary one-dimensional parameter while the index finger is used to select a primary target in two dimensions. This includes finding the number of discrete zones that a user is capable of efficiently selecting from, as well as measuring the impact of visual feedback on expert user performance. By better understanding the capabilities and limitations of using these two fingers on touch-sensitive devices, we can develop efficient interaction techniques to perform multiple simultaneous operations such as command selection and direct manipulation.

6.3.2 Apparatus

An upright 24" LCD display running at a resolution of 1024x768 pixels was used to present visual stimuli. The experimental software was run on a P4 3.0GHz PC running Windows XP.

Thumb and index finger positions were detected by placing a DiamondTouch system [Diet01] horizontally on a desk between the user and the display. Since the DiamondTouch only provides a bounding box around two or more contact positions on its surface, it is very difficult to disambiguate between fingertips. To overcome this limitation, we positioned the device so that a right-handed user's thumb would always be below and to the left of the index finger throughout the experiment. Contact information is reported at approximately 22Hz with an interpolated sensor resolution of 2752x2064 and a physical diagonal touch-surface measurement of 107cm, which allows for detecting fingertips with an accuracy of up to 0.03cm. For the experiment we only used one quarter of the physical area of the device, which provided a sensor resolution of 1376x1032. By mapping this area to the corners of the 1024x768 display, we were able to detect contact information with sub-pixel accuracy.

As depicted in Figure 6.1, the DiamondTouch was placed in front of the display and offset to the right. Participants were then seated in front of the DiamondTouch and facing towards the display so that the right hand could be placed comfortably in the active area. While many

studies on finger span from the motor control literature explicitly prevent the participant from viewing their hand, we allowed users to maintain peripheral hand visibility to better simulate the real-world user interface conditions in which we expect our designs to be used.

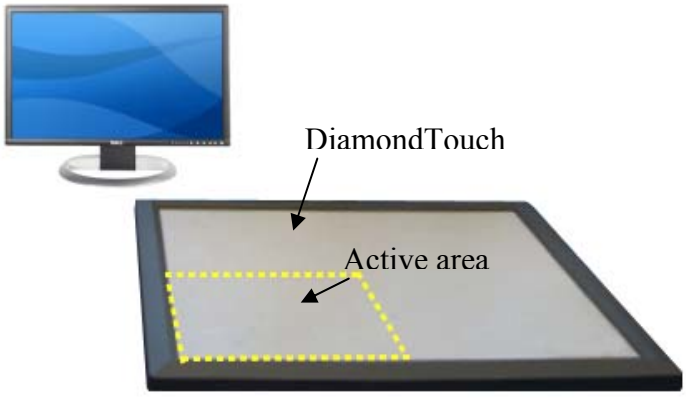


Figure 6.1 – Experimental configuration with the DiamondTouch.

6.3.3 Participants

Nine participants, two female and seven male, 20-30 years old, volunteered for the experiment. All were right-handed and had little to no experience with multi-point touch-sensitive devices. Before beginning the experiment, each participant was asked to place their thumb and index finger on the touch-sensitive surface so that their finger span was maximal but still comfortable. This maximum span was used as a calibration parameter by the software to account for different hand sizes and finger lengths.

6.3.4 Task and Stimuli

A compound target selection task was used, where the user was required to position a standard 2D cursor into a green circular target using the index finger while also selecting a discrete sub-target from a menu by adjusting the span between the thumb and index finger. Similar to Chapter 5, Figure 6.2 shows this asymmetric assignment of tasks to the two fingers, where finger span was discretized into N zones and thumb motion was used to change the active discrete selection. Targets would randomly appear in one of three fixed locations arranged as an equilateral triangle (Figure 6.3a), where the distance from any one target to the other two was 23cm in terms of the corresponding metric positions on the touch-sensitive surface. Target positions were also chosen to eliminate large finger spans from

reaching the edges of the touch-sensitive surface. We chose three target locations instead of only two so that participants had less of a chance of anticipating where the next green target would appear. This allows the difficulty of the compound selection task to remain uniform for both the 2D target selection as well as the 1D discrete sub-target selections.

Each trial began by asking the user to move the cursor into a randomly chosen target location that appeared as a circular green outline. Once the cursor was inside of the target and the span between the thumb and index finger was less than 1cm, the trial would begin. Subsequently, a new circular target would randomly appear in one of the other two remaining target positions, along with textual information inside of the circle specifying the sub-target that should be selected by adjusting finger span with the thumb. The sub-target information appeared as a fraction of the maximum finger span required to reach it (Figure 6.3b). In other words, a sub-target caption displayed as *i/N* denotes the *i*-th discrete target from a total of *N* equal-sized zones. To complete the trial, a user was required to position the cursor inside of the green target as well as select the appropriate discrete target by adjusting finger span. To confirm selection, the user tapped once with the thumb by raising it from the touch-sensitive surface and then quickly placing it back down.

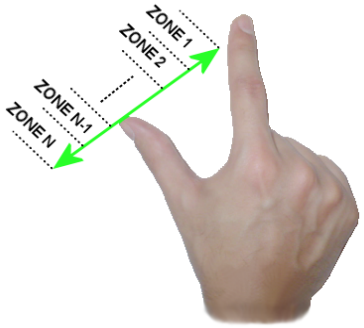


Figure 6.2 - The span between the thumb and index finger is divided into discrete zones for selecting commands or states. Thumb motion adjusts the active zone, while the index finger is used to control the position of the cursor and set the focus.

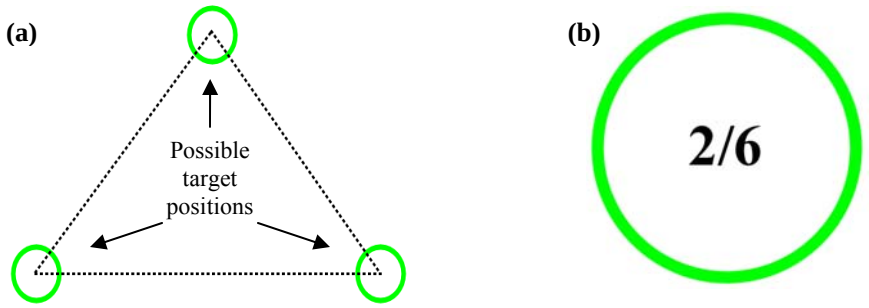


Figure 6.3 - (a) The potential target locations are arranged at the corners of an equilateral triangle; (b) An example of a target with fractional sub-target information inside of it.

Fractional information was chosen for the discrete sub-targets instead of more traditional menu items such as colors or commands since it allowed us to better estimate expert performance. Kurtenbach and Buxton used a somewhat similar approach to simulate expert behavior with hierarchic marking menus [Kurt93], where eight-item menus were labeled with compass directions instead of actual commands or colors, under the assumption that users were already familiar with such a layout. While it can be argued that fractions are not as intuitive as compass directions, the finger span studies described earlier [Ekma67, Sant97, Vand95] suggest that humans are capable of accurately estimating finger span given some mental, tactile, or visual stimuli.

We used three different visual feedback conditions for the cursor and menu, as shown in Figure 6.4. The first was *Full Visual* (*FV*), which shows the standard 2D arrow cursor along with a horizontal semi-transparent blue menu underneath it. This menu allows the user to see the number of zones that the maximum finger span range is divided into, along with continuous visual feedback about the active discrete target (Figure 6.4a). The menu always appears relative to the position of the cursor, and the entire menu also shifts left or right so that the active menu selection (based on finger span) always appears directly below the cursor. The second visualization condition was *Partial Visual* (*PV*), which only shows the active discrete target underneath the cursor location in order to reduce the amount of screen space that the widget uses (Figure 6.4b). This visualization allows users to verify the proper menu item before confirming selection. Finally, the *No Visual* (*NV*) condition only shows the cursor, without any visual feedback regarding the active menu item or menu discretization

(Figure 6.4c). Therefore, a user is forced to rely on proprioception when adjusting finger span, which simulates the condition where an expert user may want to make eyes-free selections in a manner similar to Marking Menus [Kurt93].

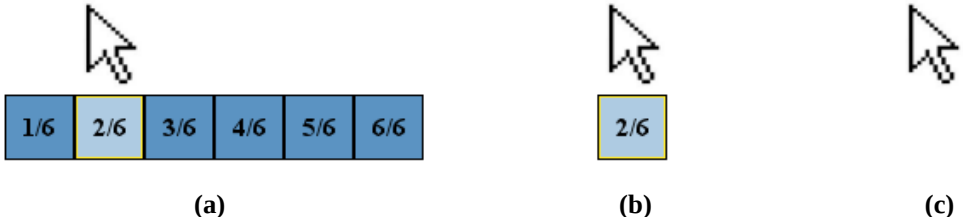


Figure 6.4 - (a) Full menu visualization; (b) Partial menu visualization; (c) No menu visualization.

6.3.5 Procedure and Design

We used a within-participants full factorial design with repeated measures. Independent variables were visualization condition (*FV*, *PV*, *NV*), the size of the menu ($N = 4, 6, 8, 10$), and the discrete sub-target that was to be selected using finger span ($S = 1, 2, 3, 4$).

Since there is no consensus as to whether humans perceive finger span as being larger or smaller than actual span, we distributed the four discrete sub-targets across the maximum finger span range so that the fractional discrete target captions represented the midpoint of each equal-sized zone for a particular menu size (see Table 6.1 and Figure 6.5).

Participants were randomly assigned to 3 groups of 3 participants each. Within each group, participants were exposed to all three visualization conditions, with the order of appearance balanced using a Latin square. For each visual feedback condition, participants completed a session of 4 blocks, where each block consisted of trials for all 16 *N*-*S* conditions, repeated 4 times in random order. In summary, the experiment design consisted of:

9 participants x

3 visualizations x

4 blocks x

4 menu sizes ($N = 4, 6, 8, 10$) x

4 discrete targets ($S = 1, 2, 3, 4$) x

4 repetitions

= 6912 total selection trials

Participants were also given a warm-up block at the start of each visualization condition in order to become familiar with the task. Participants were informed that they could take breaks between individual trials as well as between each visual feedback condition. Participants were also instructed to complete each trial as quickly and as accurately as possible, but we did not provide any specific instruction as to whether the discrete sub-targets should be selected serially or in parallel to the circular target selection with the index finger. This was intentional in order to measure whether users naturally adopt a serial or parallel strategy when completing the compound selection task. Finally, participants were instructed to keep both the thumb and index finger on the touch-sensitive surface at all times during a trial so that finger kinematics could be measured continuously. In total, the experiment lasted approximately 1 hour for each participant.

Table 6.1 - Valid span ranges (as percentages of maximum span) for each discrete sub-target.

N	***S***	**Target Caption**	**Target Caption as a Percentage**	**Valid Span Range**
4	1	1/4	25 %	12.5 – 37.5 %
	2	2/4	50 %	37.5 – 62.5 %
	3	3/4	75 %	62.5 – 87.5 %
	4	4/4	100 %	87.5 – 112.5 %
6	1	2/6	33.3 %	25 – 41.6 %
	2	3/6	50 %	41.6 – 58.3 %
	3	4/6	66.7 %	58.3 – 75 %
	4	5/6	83.3 %	75 – 91.6 %
8	1	2/8	25 %	18.75 – 31.25 %
	2	4/8	50 %	43.75 – 56.25 %
	3	5/8	62.5 %	56.25 – 68.75 %
	4	7/8	87.5 %	81.25 – 93.75 %
10	1	3/10	30 %	25 – 35 %
	2	5/10	50 %	45 – 55 %
	3	7/10	70 %	65 – 75 %
	4	9/10	90 %	85 – 95 %

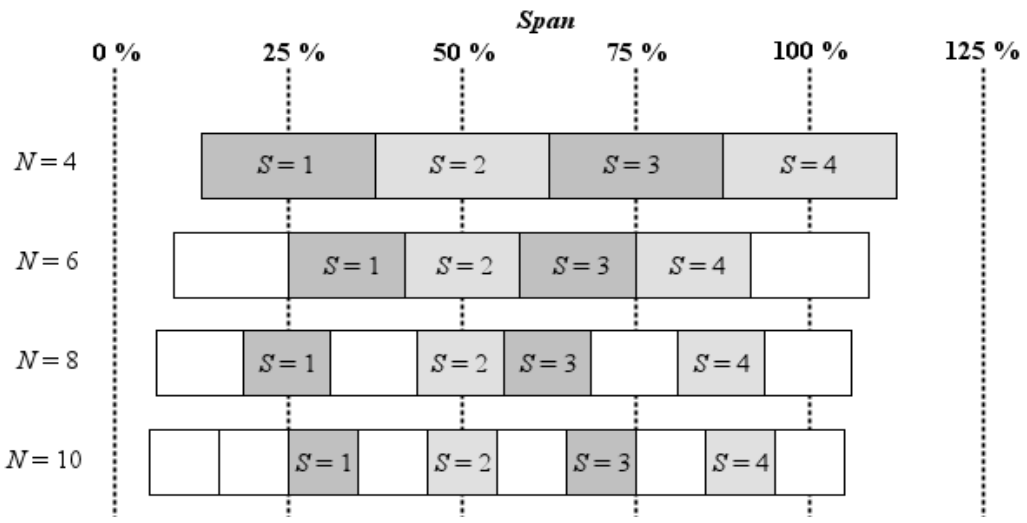


Figure 6.5 - Distribution of the discrete targets across the finger span range for each of the menu sizes used in the study.

6.3.6 Results

Dependent variables were *movement time* (*MT*), which is defined as the time it takes from the start of a trial to when the cursor is inside of the circular target and the thumb is tapped for the first time; *completion time* (*CT*), which is the time it takes from the start of a trial to when a successful selection is made or too many incorrect selections are performed; *error rate* (*ER*), which represents the percentage of trials for a particular *N*-*S* combination that are selected incorrectly; *parallelism* (*P*), which denotes the percentage of time during a trial that both the position of the index finger as well as the span between the thumb and index finger were being adjusted concurrently in any direction; *inefficiency* (*I*), which measures the trajectory of the index finger's path against the shortest path from one circular target to another during a trial; and *number of crossings* (*NC*), adapted from Ramos et al. [Ramo04], which represents the number of times the thumb enters or leaves the correct discrete target for a trial before the selection is confirmed.

Incorrect selections were defined as those which were confirmed with a thumb tap when the cursor was inside of the circular green target and the incorrect discrete target was selected with finger span. These incorrect selections resulted in an audible beep, along with a textual message informing the user to try again. Participants were allowed to make up to five incorrect selections per trial, after which point the next trial would begin. Selections which

were confirmed with a thumb tap while the cursor was outside of the circular green target also resulted in an audible beep and textual message, but they were not counted as actual selections.

6.3.6.1 Movement Time Analysis

Since selecting discrete sub-targets by adjusting finger span with the thumb is essentially a one-dimensional pointing task along a fixed-size axis, Fitts' Law [Fitt54] tells us that increasing the number of zones (which is equivalent to making each zone smaller) will increase the time it takes to select a target. As expected, analysis of variance showed a significant effect for both the number of menu items (*N*) and the finger span required for each sub-target (*S*) on movement time ($F_{3,24}$=54.18, p<0.01 and $F_{3,24}$=21.86, p<0.01 respectively), with sub-targets that were smaller and farther away from the minimum span taking longer to select.

One of the main goals of this experiment was to assess the impact of visual feedback on the performance of the discrete target selections. A repeated measures analysis of variance showed no significant difference between the three visual feedback conditions on *MT*. However, there was a significant visualization x *N* interaction for *MT* ($F_{6,48}$=5.03, p<0.01). Figure 6.6a plots the average movement time for each of the menu sizes under the various visual feedback conditions, with *NV* taking less time than the *PV* and *FV* conditions as the number of menu items increases.

The block number was found to have a significant effect on *MT* ($F_{3,24}$=10.27, p<0.01), with a pairwise means comparison showing a significant difference between the first and last blocks (p<0.01). There was no significant visualization x block interaction, however, which suggests that the learning effect was consistent for all three visualization conditions.

6.3.6.2 Completion Time Analysis

Completion time (*CT*) allows us to determine the overall amount of time spent in a trial up until a correct selection is made or five incorrect selections are made. This contrasts with *MT,* which only represents the amount of time in a trial up until the first selection (correct or incorrect). A repeated measures analysis of variance showed a significant effect of visual

feedback condition on *CT* ($F_{2,16}$=15.39, p<0.01), with mean completion times of 2.66, 2.69, and 3.33 seconds for the *FV*, *PV*, and *NV* conditions respectively. Pairwise means comparison showed no significant difference between the *FV* and *PV* conditions, but there was a significant difference between *FV* and *NV* (p<0.01) and *PV* and *NV* (p<0.01). This result differs considerably compared to the *MT* results. Figure 6.6b plots the average *CT* for the various menu sizes under the different visualization conditions.

The number of menu items (*N*) and the finger span required for a sub-target (*S*) had a significant effect on *CT* ($F_{3,24}$=143.78, p<0.01 and $F_{3,24}$=18.43, p<0.01 respectively), which is similar to the results we found with *MT*. A significant visualization x *N* interaction on *CT* ($F_{6,48}$=22.98, p<0.01) was also found, with a pairwise means comparison showing significant differences between *FV* and *NV* (p<0.01) and *PV* and *NV* (p<0.01) for *N* = 6, 8, and 10. However, no difference in *CT* was found between *FV*, *PV*, and *NV* for *N* = 4.

Block number also had a significant effect on *CT* ($F_{3,24}$=6.23, p<0.01). However, there was no significant visualization x block interaction, which is similar to the results we found with *MT*.

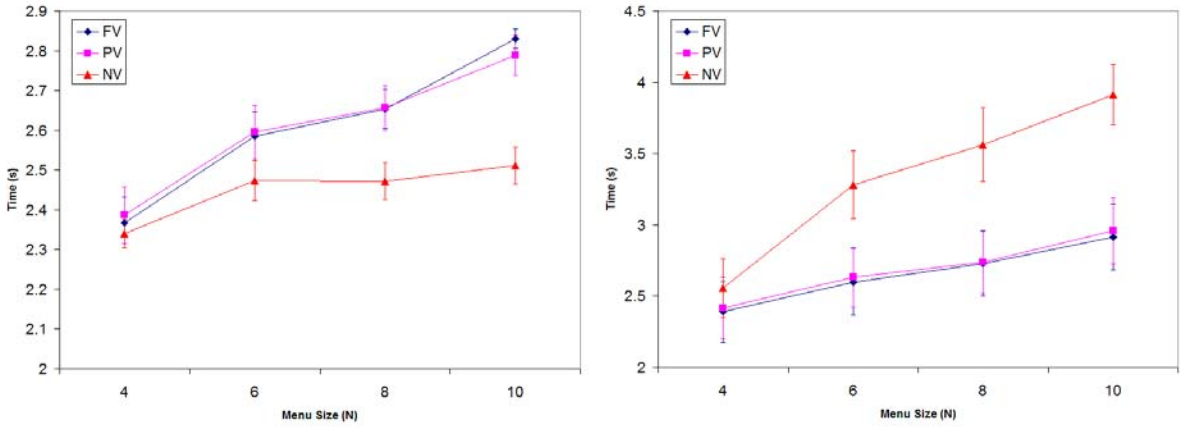


Figure 6.6 - Effect of visual feedback and menu size on (a) movement time (*MT*); (b) completion time (*CT*). Error bars show standard error.

6.3.6.3 Error Rate Analysis

A repeated measures ANOVA indicated that menu size (*N*) had a significant effect on *ER* ($F_{3,24}$=91.51, p<0.01), with mean error rates of 5.4%, 16.8%, 22.8%, and 26.6% for *N* = 4, 6,

8, and 10 respectively. However, there was no significant effect of the target finger span (*S)* on *ER*.

Visual feedback condition also had a significant effect on *ER* ($F_{2,16}$=138.38, p<0.01). Pairwise means comparison showed significant difference between all pairs of visualizations (p<0.05). Overall, error rates were 4.1%, 6.8%, and 41.2% for *FV*, *PV*, and *NV* respectively.

A significant visual feedback x menu size interaction for *ER* ($F_{6,48}$=49.98, p<0.01) indicates that the *FV* and *PV* conditions result in error rates of less than 12% for all menu sizes up to 10 items, but that the *NV* condition results in error rates of over 40% with 6 or more menu items. Menus with 4 items, however, resulted in less than 6% errors for the *NV* condition. Figure 6.7 plots the error rates for the various menu sizes under the three visual feedback conditions.

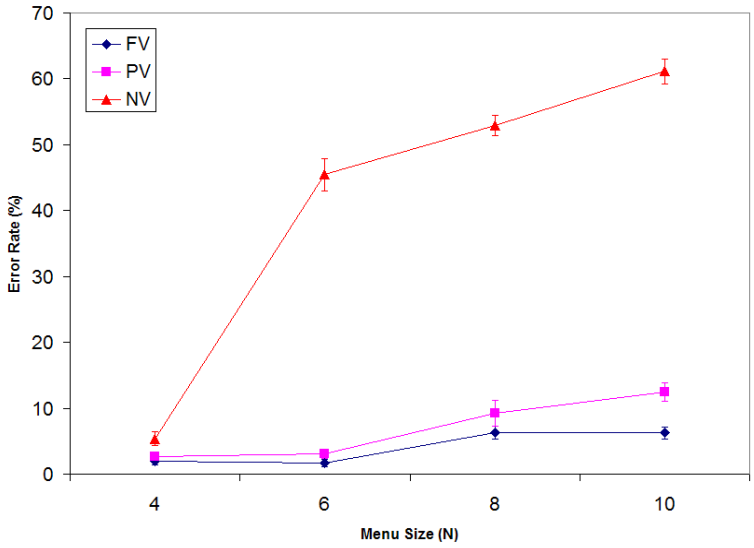


Figure 6.7 - Effect of visual feedback and menu size on error rate (*ER*) with standard error bars.

In terms of learning, there was no significant effect of block number on error rate. Additionally, there was no visual feedback x block interaction for *ER* which suggests that learning effects were consistent across each of the visualizations. This is particularly interesting for the *NV* condition, since it suggests that users were not getting any better at selecting discrete targets using proprioception alone.

Similar to studies from the motor control and psychophysics literature, we also wanted to determine whether participants overestimated or underestimated the discrete sub-targets using finger span. Figure 6.8 plots the average span that was selected for each of the different menu items across all trials for the *NV* condition only. Results suggest that participants made large positive errors (span > target) for small fractional sub-targets, but larger sub-targets resulted in negative errors (span < target). The consistent negative errors for the fractional sub-targets above 75% could be attributed to participants not adjusting their span beyond the maximum comfortable span that was set during the calibration phase. Based on the plot, actual span appears to be within 15% of the target span. The equation of the regression line is y=17.881 + 0.747x, with r^2=0.973. Overall, these results represent a middle ground between the conflicting studies described earlier, which suggests to us that the type and familiarity of the stimulus presented to the subject plays an important role. Our results may also slightly differ from previous results due to our focus on a compound selection task as well as allowing peripheral vision of the hand for the *NV* condition.

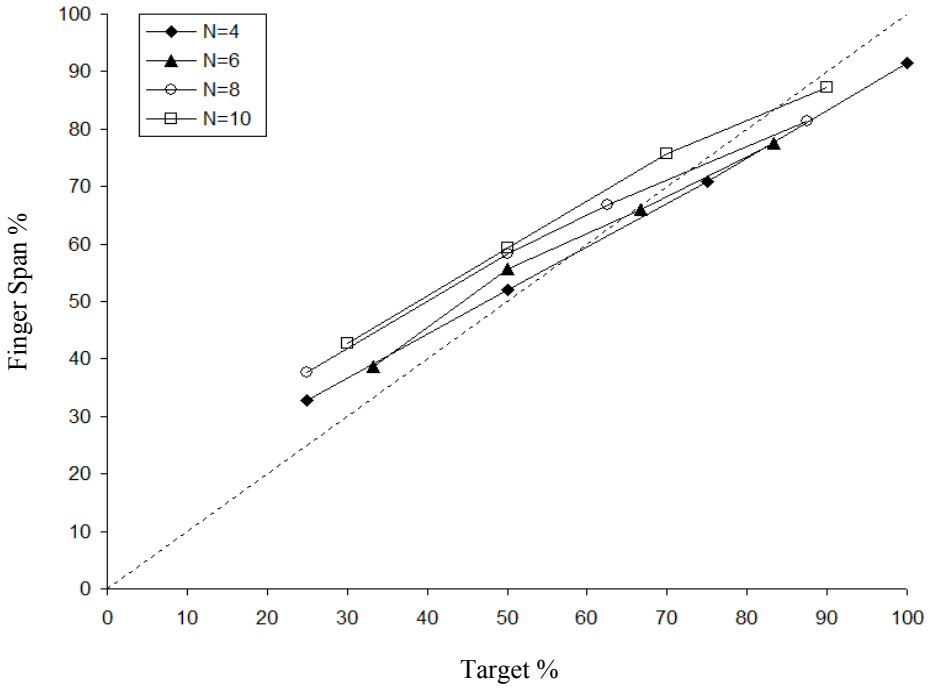


Figure 6.8 - Matching finger span to fractional targets in the *NV* condition. The dashed line represents a one-to-one relationship between finger span and the fractional sub-targets for comparison. Data points represent the average span across all trials when each target was first selected.

6.3.6.4 Crossing Analysis

The number of crossings (*NC*) provides us with a simple measure of the amount of control users have when selecting sub-targets with finger span. A perfect selection would result in an *NC* value of 1, while higher values of *NC* would suggest that users are searching around with their thumb before settling on some particular span. One potential problem with the *NC* metric occurs when users never enter the correct sub-target, which results in an *NC* value of zero. In such instances, we replaced the zero *NC* value with the mean *NC* from all trials for the particular participant as a penalty.

Analysis of variance showed a significant effect of *N* on *NC* ($F_{3,24}$=86.07, p<0.01), with pairwise means comparisons showing significant difference between all menu sizes (p<0.01). On average, the number of crossings were 1.13, 1.25, 1.63, and 1.96 for N = 4, 6, 8, and 10 respectively.

The number of crossings was also significantly different across *S* ($F_{3,24}$=32.35, p<0.01). Menu items that required a smaller span resulted in more crossings, while menu items closer to the maximum span resulted in fewer crossings. The average number of crossings were 1.72, 1.61, 1.45, and 1.17 for S = 1, 2, 3, and 4 respectively. This suggests that spans closer to the maximal span were selected with more confidence than those closer to the minimal span.

Visual feedback also significantly affected *NC* ($F_{2,16}$=6.08, p<0.01). The average number of crossings was 1.60, 1.52, and 1.32 for *FV*, *PV*, and *NV* respectively. This suggests that in the *NV* condition users were either very confident in their chosen span, or they were completing selections based on blind faith.

Similar to *MT* and *ER*, there was a significant visual feedback x menu size interaction for *NC* ($F_{6,48}$=8.01, p<0.01). As shown in Figure 6.9, *NC* was consistently lower for the *NV* condition with menu sizes of 6, 8, and 10, while 4 menu items resulted in *NC* values that were almost identical across all three visualizations.

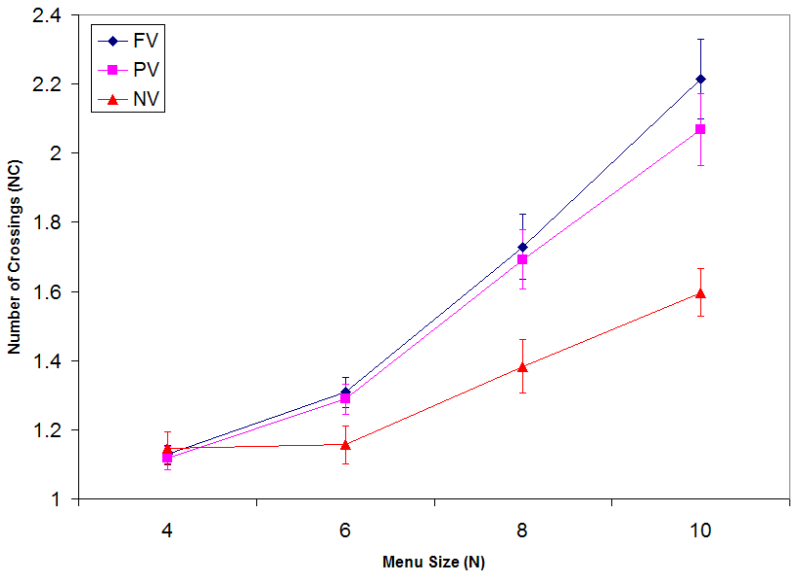


Figure 6.9 - Effect of visual feedback and menu size on the number of crossings (*NC*) with standard error bars.

6.3.6.5 Inefficiency Analysis

Zhai proposed the following formula for computing the inefficiency of multiple degree of freedom motion [Zhai98]:

$$\frac{\text{Length of actual path - Length of shortest path}}{\text{Length of shortest path}}$$

We computed this metric for each trial to measure the inefficiency (*I*) of the index finger's 2D trajectory from one circular green target to another. This metric was chosen over the efficiency (*EFF*) metric from Chapter 5 since the latter requires a measure of the total error reduced for a particular trial, which can only be computed when a trial is completed successfully. Since our current experiment allows incomplete trials, we opted for the inefficiency metric instead. Averaged across all trials, *I* was 0.18, which indicates that the chosen trajectories were well-controlled and close to being optimal. While this measure alone is not very interesting for simple 2D pointing tasks with the index finger, it can give us some insights into the coordination of the compound nature of our task as discussed in the following section.

6.3.6.6 Parallelism Analysis

Since our compound selection task required participants to perform two separate but arguably dependent selections, we wanted to assess the amount of parallelism (*P*), if any, that was

exhibited. Averaged across all trials, participants performed the compound selections in parallel 43.9% of the time. This is interesting since we did not explicitly tell users that the two subtasks could be performed in parallel. Building on our results from Chapter 5, this high degree of parallelism suggests that using the thumb and index finger in this asymmetric manner is natural and does not present significant motor or cognitive difficulty. Additionally, the low average number of crossings for sub-target selections as well as the low average inefficiency for the index finger's motion indicates that parallel movements were intentional and well-controlled.

Analysis of variance showed a significant effect of menu size (*N*) and visualization condition on parallelism ($F_{3,24}$=6.63, p<0.01 and $F_{2,16}$=4.66, p<0.05 respectively). There was no significant visual feedback x menu size interaction on *P*. Figure 6.10 plots the percentage of parallelism for the different visual feedback conditions and menu sizes, with results showing that parallelism was consistently higher in the *NV* condition, followed by *FV* and *PV*.

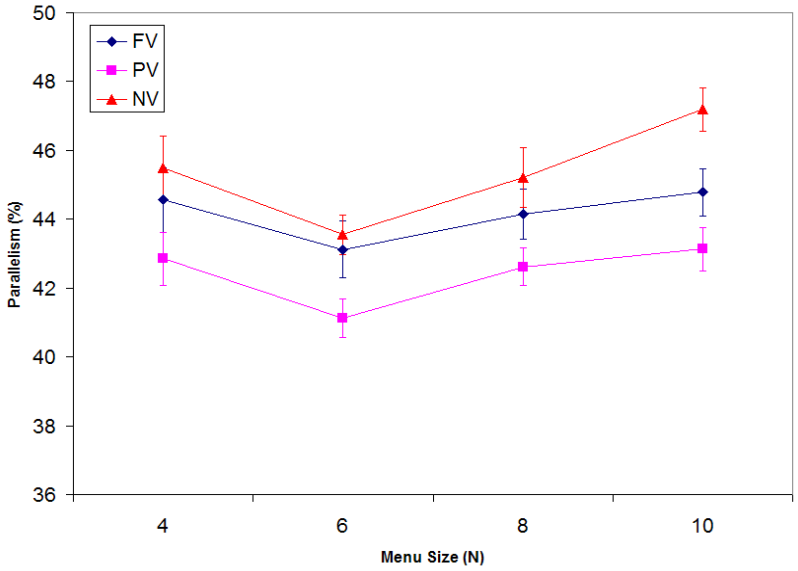


Figure 6.10 - Effect of visual feedback and menu size on parallelism (*P*) with standard error bars.

6.3.7 Discussion

The experimental results suggest that without any visual feedback, users can effectively select from up to 4 menu items using finger span, with performance that is equal to or better than the partial and full visual feedback conditions. However, with more than 4 items, error

rates reach unacceptable levels without any visual feedback. Under partial and full visual feedback conditions, however, results suggest that users can select from all of the tested menu sizes with acceptable performance. Most interestingly, performance under partial and full visual feedback was comparable, which indicates that once a user has a mental image of the relative positions of the various commands on a menu, the amount of screen-space used by the *FV* menu can be reduced to the *PV* menu without significantly reducing performance. Another interesting finding was the consistently lower *MT* for the *NV* condition across all menu sizes. This suggests that users very quickly made an estimate of the required finger span and tried to confirm the selection when there was no visual feedback, whereas with the *FV* and *NV* conditions users waited for visual confirmation of the correct menu item which took more time as menu size increased. However, the analysis of *CT* (which also takes incorrect selections into consideration) shows that for large menu sizes the actual time spent in a trial was consistently higher for the *NV* condition. Clearly, the *MT* results must not be interpreted in isolation but rather in conjunction with the *ER* results. Nevertheless, if error rates could somehow be brought down, the *NV* condition may in fact be highly efficient even in terms of *CT*. Such improvements could possibly be made by better understanding the span perception curve (Figure 6.8) so that discrete targets could be more efficiently distributed across the range of finger spans in order to maximize accuracy for higher menu sizes. We also imagine accuracy improving with direct-touch displays under the *NV* condition, since users may be able to use the direct vision of their hand as an additional cue to estimate finger span.

It is important to note that, although the precision of the DiamondTouch was acceptable for the purposes of our experiment, it can be argued that accuracy may improve in the *NV* condition by using higher-precision touch-sensitive surfaces. If this is the case, and we assume that new input devices will provide accuracy that is at least as good as that provided by the DiamondTouch, an interface designer can consider our results as an acceptable lower bound on expected performance. However, considering that the DiamondTouch can discriminate contact points at a resolution that is roughly over 30 times smaller than the tip of the "average" adult finger, we don't expect significant improvements in accuracy with higher-precision devices.

Finally, our experiment assumed that the multi-point input device was capable of differentiating between the thumb and index finger. While many existing multi-point devices (including the DiamondTouch) do not automatically support this at the hardware level, a simple approach was proposed by Wu and Balakrishnan [Wu03] where labels were assigned based on the order of touch detection. Nevertheless, the Visual Touchpad (Chapter 3) and the SmartSkin [Reki02] have demonstrated effective finger labeling, and it seems reasonable to expect this to become a standard feature on multi-point surfaces of the future. Additionally, we expect our span perception results to be applicable to techniques that use the thumb and index finger in a symmetric manner as discussed in Chapter 5, where correct finger labeling is not critical.

6.4 Widget Design Variations

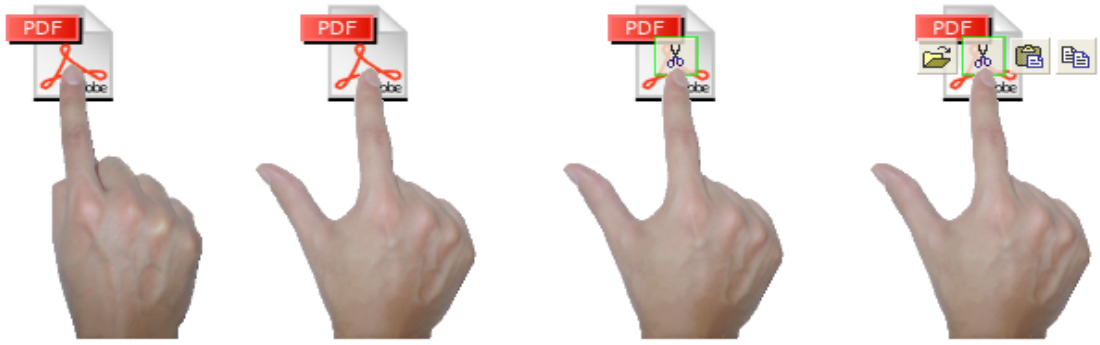
Our experiment demonstrated that users are capable of positioning the index finger in 2D while the thumb is used to adjust finger span asymmetrically to select a discrete parameter. Based on the results of the experiment, we designed the following three practical menu widgets that build upon the basic menu design used in the study.

6.4.1 Self-revealing ThumbToolglass

The bimanual toolglass, first proposed by Bier et al. [Bier93], allows a user's non-dominant hand to position a toolbar item directly over top of a target so that the dominant hand can "click-through" the toolbar item onto the target object, thereby merging command selection and direct manipulation. This combining of tasks into a single operation has been shown to improve performance over standard tool palettes or pop-up menus [Guim05].

The basic menu design used in our experiment can also be used in such a manner by placing the center of the active menu item at the cursor position (instead of underneath it) so that selections can be confirmed by tapping the index finger instead of the thumb. This effectively allows a toolglass to be manipulated with a single hand, where all of the menu items are arranged horizontally in a single row and commands are chosen by adjusting finger span with the thumb. Since our experimental results suggest that users are capable of selecting up to 4

menu items without any visual feedback of the menu layout, we have designed a *Self-revealing Single-handed Toolglass* that allows expert users to make compound selections in an eyes-free manner, while novice users receive sufficient information about the menu in order to become accustomed to the arrangement of commands. This is similar to the approach taken with Marking Menus, where expert users can make quick strokes once the menu is activated without actually seeing the entire menu on the screen [Kurt93]. Figure 6.11 shows a 4-item toolglass with open, cut, copy, and paste commands being used on a direct touch-sensitive display. As shown in Figure 6.11b, the toolglass becomes active as soon as both the thumb and index finger are detected on the touch-sensitive surface. However, no visual feedback is shown immediately. Therefore, an expert user can complete the compound selection in a gestural, eyes-free manner by quickly setting the appropriate finger span with the thumb and then tapping with the index finger onto the desired object. If no selection is made within 500ms, the active command associated with the current finger span is shown beneath the index finger (Figure 6.11c). This allows a user with only a basic mental image of the toolglass layout to receive some visual feedback in order to confirm the desired command selection, with minimal screen space usage. If the user still does not make a selection after an additional 500ms, the system assumes the user is unfamiliar with the menu layout and the entire toolglass fades into view (Figure 6.11d). The state machine for this self-revealing toolglass is shown in Figure 6.12.



(a) **(b)** **(c)** **(d)**

Figure 6.11 - A Self-revealing Single-handed Toolglass being used on a direct multi-touch display. (a) The single index finger controls cursor position only; (b) The thumb activates the toolglass immediately so that expert users can complete selection without visual feedback; (c) After 500ms of inactivity, partial visual feedback is provided by showing the active toolglass item based on current finger span; (d) After another 500ms of inactivity, the entire toolglass fades into view for novice users.

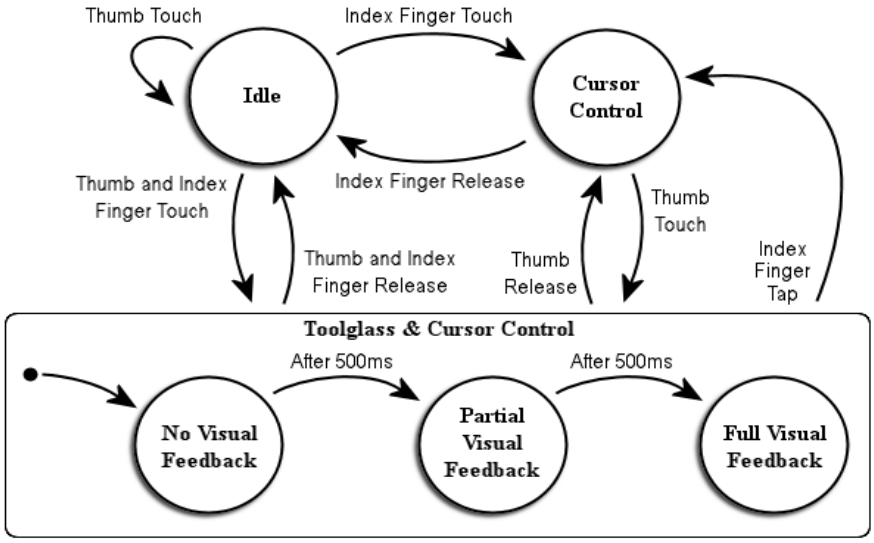


Figure 6.12 - Hierarchic state machine for the Self-revealing Single-handed Toolglass.

Our experimental results suggest that for menus with more than 4 items the *NV* condition will lead to unacceptably high error rates. Therefore, for applications which require more than 4 items on a toolglass, the *No Visual Feedback* state should be omitted so that the toolglass immediately transitions into the *Partial Visual Feedback* state. This approach provides the benefit of reducing the amount of screen real estate that the toolglass occupies, while also allowing novice users to become accustomed to the layout of commands. However, expert usage is not completely gestural and eyes-free as in the 4-item menu.

As demonstrated by Bier et al. [Bier93], combining command selection and direct manipulation in this manner is beneficial since it reduces the frequent focus changes that are required with standard serial pop-up menus or distant toolbars. While certain techniques such as Marking Menus, FlowMenus, and control menus also allow for merging command selection and direct manipulation, they require temporarily moving away from the point where the command was invoked, which may cause difficulties with manipulations such as free-form drawing [Guim05]. Both bimanual toolglasses and our Self-revealing Single-handed Toolglass do not have this problem.

6.4.2 Bi-digital Marking Menus

One of the shortcomings of the Single-handed Toolglass is the small number of menu items that can be selected in an eyes-free manner. In comparison, hierarchic Marking Menus allow for up to 3 depth levels with 8 menu items at each level to be selected in an eyes-free manner with low error rates [Kurt93]. This, however, does not render our single-handed toolglass obsolete.

With single-point touch-sensitive devices, activating a Marking Menu is challenging. On a multi-point input device, however, two fingers can be used to denote menu activation. If we assume that these two fingers are the thumb and index finger from the same hand, we propose using a Self-revealing Single-handed Toolglass to both activate as well as extend the capabilities of a Marking Menu.

Since finger span can be discretized into a one-dimensional array of zones, we can imagine each of these zones to represent an extra "layer" at the first level of a Marking Menu. Therefore, with a discretization of finger span into just 4 zones, an 8 directional single-level marking menu can be combined with a single-handed toolglass to allow up to 32 menu options. Figure 6.13 demonstrates such a *Bi-digital Marking Menu* combined with a 3-item single-handed toolglass.

For a novice user, the Marking Menu can be activated by outstretching and holding the thumb for 500ms. The active pie menu associated with the current finger span then appears directly underneath the index finger. After another 500ms, the other potential pie menus appear in a reduced form beside the active menu (Figure 6.13a). Therefore, by adjusting finger span, the pie menus can be shifted left or right in order to place the desired menu underneath the index finger. To complete selection, the user could then simply draw a stroke into the direction of the desired menu option (Figure 6.13b). From the perspective of an expert user, the command selection can be completed in a gestural, eyes-free manner by simply outstretching the thumb to the desired finger span followed by a quick stroke in the appropriate direction (Figures 6.13c and 6.13d).

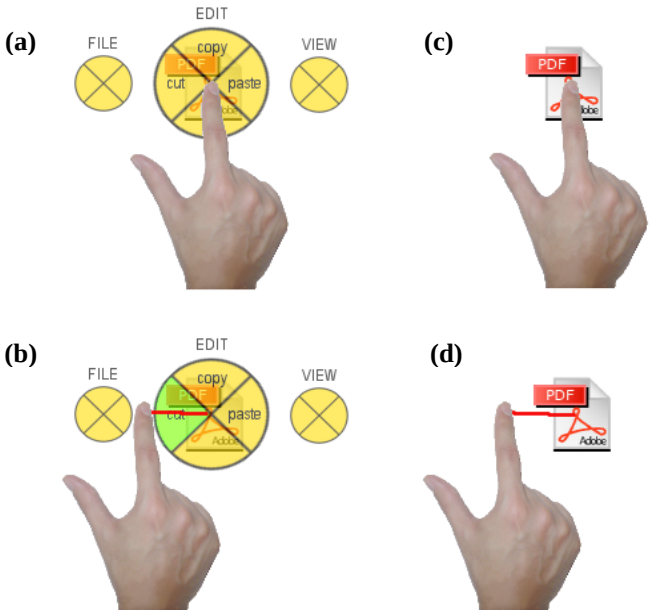


Figure 6.13 - Bi-digital Marking Menus are activated by the detection of two fingers, while different menus can be brought into focus based on finger span. (a-b) Novice users see the entire set of marking menus after holding the thumb for at least 500ms; (c-d) Expert users can immediately initiate the selection stroke without any visual feedback of the pie menus.

Although it is beyond the scope of this paper, a promising direction for further research would be to determine whether there are any performance benefits to multiplexing the first level of a Marking Menu in this manner versus increasing the number of depth levels as is normally the case.

6.4.3 Multi-finger Chorded Toolglass

Since a small but growing number of multi-point touch-sensitive devices are capable of detecting all five fingers from each hand [Reki02, Han05], we can extend the capabilities of our toolglasses with the additional contact information from the middle, ring, and little fingers. Since the motor control literature suggests that these other three fingers exhibit significant enslaving effects when attempting to control them independently [Hage00], we have decided to use them only as binary modifiers. As in the bi-digital toolglass, the index finger always controls the cursor position while the span between the thumb and index finger controls the active discrete menu item (Figure 6.14a). However, when the system also detects different combinations of the other three remaining fingers making contact with the touch-

sensitive surface, the set of items on the toolglass can be changed (Figure 6.14b-d). This approach allows the number of items on the bi-digital toolglass to be scaled up reasonably well. Unlike the previous two designs, however, these *Multi-finger Chorded Toolglasses* are not as self-revealing. Therefore, it would be interesting to investigate visualization techniques that allow novice users to more easily discover which items are associated with the various finger combinations without physically forming them.

Figure 6.14 - A Multi-finger Chorded Toolglass assigns different tooglasses to the thumb and index finger based on combinations of the middle, ring, and little fingers. (a) Standard single-handed toolglass controlled with the thumb and index finger; (b) Middle finger activates a toolglass with editing commands; (c) Middle and ring finger combine to activate a toolglass of drawing tools; (d) Middle, ring, and little finger together enable a selection toolglass.

6.5 Summary

Using the span between the thumb and index finger of a single hand was shown to be a viable approach for combining direct manipulation and command selection on multi-point touch-sensitive surfaces. Our user study showed that the performance of using finger span for selecting discrete commands depends significantly upon the number of items as well as the amount of visual feedback presented to the user. Additionally, our results suggest that users are quite capable of performing compound tasks when separate but dependent roles are assigned to the thumb and index finger. Based on the results of the experiment, we presented three practical widget designs that allow for merging command selection and direct manipulation while at the same time facilitating smooth transitioning from novice to expert usage.

Chapter 7

Conclusion

7.1 Summary

This thesis explored a number of open issues related to multi-finger interactions on touch-sensitive surfaces. In Chapter 3 we first addressed some of the open issues from an input device perspective by developing the Visual Touchpad, a low-cost vision-based input device that allows for detecting multiple hands and fingertips over a constrained planar surface. While the accuracy at which the prototype device provides contact information is currently limited by camera resolution and processing power, the system demonstrates how more detailed information about the fingers such as labels, orientation, and hover can be detected. To the best of our knowledge, there is currently no other multi-point device that can extract all of these features simultaneously.

In Chapter 4 we designed and implemented three interactive multi-finger systems: a fluid multi-finger picture manipulation application, a multi-finger system for interacting with large displays from a distance, and an interactive art installation for the Deaf Culture Centre in Toronto that demonstrates the expressiveness of multiple hands and fingers. These system designs served two major purposes. First, they allowed us to demonstrate how the low-cost Visual Touchpad from Chapter 3 could actually be used in real-world user interfaces instead of relying on more expensive and elaborate tracking systems. Second, they allowed us to

perform some initial explorations into how multiple fingers could be used for performing high degree-of-freedom manipulations.

The work presented in Chapter 5 then took a step towards gaining a better understanding of how two fingers from a single hand could be used effectively for high degree-of-freedom input. In particular, we investigated how the thumb and index finger, which are arguably the two most important fingers of the human hand, could be used to enhance existing single-finger interaction techniques. We proposed a fluid interaction style that uses the thumb and index finger of a single hand in an asymmetric-dependent manner to control *bi-digit widgets*, where the index finger performs the primary and more frequent 2D tasks and the relative position of the thumb performs secondary and less frequent tasks to support the index finger's manipulations. We validated this interaction style by comparing it with two alternative finger mappings, and we proposed a set of design guidelines that allow a designer to determine the suitability of our approach to different tasks. We also presented a variety of bi-digit widget designs that demonstrate the capabilities and effectiveness of this interaction style.

In Chapter 6 we then continued our investigation of the thumb and index finger by assessing the impact of visual feedback on the perception of finger span when using discrete bi-digit widgets. Results suggested that users were capable of selecting from up to 4 discrete commands with the thumb without any visual feedback, which helped us to design a set of more advanced bi-digit widgets that showed how command selection and direct manipulation could be merged into a single fluid operation, with smooth transitioning from novice to expert usage.

7.2 Future Work

While this work has addressed some of the open issues related to multi-finger interactions from an input device, design, and human factors perspective, there are clearly a number of unanswered questions remaining.

From a device perspective, consistently reliable disambiguation of hands and fingers is still an unsolved problem. In the ideal scenario, a touch-surface that can detect multiple fingerprints in real-time at varying heights above the surface would be desirable, since this would allow not only perfect hand and finger disambiguation, but also orientation detection and support for any number of users. The work by Sugiura and Koseki [Sugi98] is one step in this direction, where a fingerprint scanner assigns different commands or objects to different fingertips. Due to scanning hardware limitations, however, it is still not possible to extract orientations, detect hover, or perform real-time direct manipulations.

From a human performance and design perspective, there are still many unresolved issues as to how additional finger parameters can be used effectively. Much like the work we did in Chapters 5 and 6 related to finger span, it would be interesting to look at how absolute or relative hover and pressure could be used in bi-digit scenarios for high degree-of-freedom manipulations. Similarly, investigating techniques that make use of the remaining three fingers of the hand to complement the proposed bi-digital manipulations seems like a fruitful direction for further research.

One important issue that arises with respect to the bi-digit widgets from Chapters 5 and 6 is how to integrate the techniques into a cohesive system. For example, while each of the widgets open up a number of possibilities for performing compound operations in an isolated manner, it is important to consider how a variety of widgets can be used together, and how one might switch between them in a real-world application. One possibility is to make use of the remaining three fingers of the hand, much like we did in Chapter 6 for the Multi-finger Chorded Toolglass, where the extra fingers denote different bi-digit widgets for various operations. It would also be beneficial to allow a user to customize which widgets are associated with the various finger combinations.

While the use of the remaining three fingers in this way may potentially allow the system to be integrated into more complex applications, it introduces the problem of discoverability. For example, aside from a guided tutorial or trial-and-error, the Multi-finger Chorded Toolglass does not currently provide any mechanism that allows a user to easily determine

which finger combinations are associated with the various toolglasses. Similarly, all of the bi-digit widgets proposed in Chapter 5 also suffer from a lack of discoverability to some extent, since the user must first be familiar with the role of the thumb and how to confirm selections. The modified ThumbToolglass and the Bi-digital Marking Menu from Chapter 6 took a step towards improving discoverability since they both addressed the issue of self-revelation. Another potential approach to improving discoverability is through pop-ups or textual cues that attempt to guide a user based on their behaviour. However, detecting user confusion or hesitation is a difficult problem on its own, and any approach has the possibility of introducing false activations. Therefore, this is an interesting direction for further work.

For multi-point touch surfaces to become ubiquitous, it is important to investigate techniques that allow for efficient text entry. While the virtual keyboard as presented in Chapter 4 is a reasonable approach due to its familiar layout, the lack of tactile feedback is a serious limitation that may increase error rates. However, since a multi-point surface can determine exactly where inside of a virtual key a letter was struck, a possible enhancement would be to adaptively adjust key positions to better accommodate a user's typing behaviour similar to the work by Himberg et al. [Himb03]. For hand-held devices, however, presenting a full-size virtual keyboard may not be desirable. In such scenarios, single-finger techniques such as Quikwriting [Perl98] may be appropriate, but it would be interesting to investigate alternative text-entry approaches that make use of the higher bandwidth input capabilities of two or more fingers.

The system designs from Chapter 4 all rendered a live video of a user's hands directly onto an upright display. It would be interesting to determine if these video hands have any effect on the performance of multi-finger manipulations when compared to simply showing fingertip positions or even a simple arrow cursor. Graham and MacKenzie [Grah96] compared physical pointing tasks to virtual pointing tasks and found no difference in the initial movement times, but they did find that a user's ability to close-in on a target using a virtual representation of a finger was slower than with a real finger. It's not clear that this directly applies to our system designs from Chapter 4, however, since our rendered hands are neither real nor virtual in the abstract sense, but rather an accurate visual proxy of a user's

real hands. Therefore it would be useful to more formally investigate the value of using such live hand images as is done in our designs. In a related manner, Kirsh and Maglio [Kirs94] argued that certain cognitive and perceptual tasks are better solved by doing things in the real world as opposed to solving them mentally. For example, they describe the frequent translations and rotations that users perform on Tetris pieces as an example of users trying to gain a better understanding of the situation of the entire puzzle. Such *epistemic* actions can thus be used to uncover information about a problem that may be hard for a person to understand or solve completely in the head [Kirs94]. It is worth investigating whether or not showing a user's actual hands on the screen promotes or facilitates such epistemic actions in addition to providing a compelling experience for the user.

As mentioned in Chapters 5 and 6, our bi-digit widget designs could be used in both the left and right hands in order to complement existing bimanual techniques. While Hager-Ross and Schieber [Hage00] found no differences in the amount of enslaving effects during single-finger flexion and extension tasks, Reilly and Hammond [Reil04] found that two-fingered force production tasks resulted in significantly higher independence for the dominant hand versus the non-dominant hand. Based on these findings, it would be worthwhile to determine if users have better control of our bi-digit widgets with their dominant hand which may allow us to develop a set of guidelines that allow a UI designer to assess the viability of a particular widget for non-dominant hand usage.

Finally, it would be interesting to look at how our single-handed bi-digital techniques compare with bimanual techniques that use the index finger of each hand to perform manipulations. A recent study by Moscovich [Mosc07] found that symmetric bi-digital input performed as well as symmetric bimanual input during a direct manipulation docking task. However, users perceived the task differently depending on the type of input. In the bimanual case, the docking task was perceived as the direct manipulation of two control points, while the bi-digital case was perceived as a unified position, orientation, and scale adjustment task. In a similar manner, it would be interesting to investigate how our asymmetric widget designs compare to existing asymmetric bimanual techniques.

Appendix A – Ethics Consent Form

CONSENT FORM

I agree to participate in a study that is comparing the usability of various user interface techniques on large scale displays. I understand that my participation is entirely voluntary.

The following points have been explained to me:

1. The purpose of this research is to compare human ability to use various new user interface techniques when interacting with large scale displays. I understand I will be asked questions about my previous computer experience. The benefits I may expect from the study are: (a) an appreciation of research on user interfaces, (b) an opportunity to contribute to scientific research.

2. The procedure will be as follows: During a single session lasting approximately 1 hour (including breaks), I will perform various computer interface tasks on a large scale computer display.

3. The researchers do not foresee any risks to me for participating in this study, nor do they expect that I will experience any discomfort or stress.

4. I understand that I may withdraw from the study at any time.

5. I understand that I will receive a copy of this consent form.

6. All of the data collected will remain strictly confidential. Only people associated with the study will see my responses. My responses will not be associated with my name; instead, my name will be converted to a code number when the researchers store the data.

7. The experimenter will answer any other questions about the research either now or during the course of the experiment. If I have any other questions or concerns, I can address them to the research director**,** Prof. Ravin Balakrishnan of the Department of Computer Science. He can be contacted by phone: 416-978-5359 or email: ravin@cs.toronto.edu. Directions to his office can be found on his website: www.dgp.toronto.edu/~ravin

8. Upon completion of my participation, I will receive an explanation about the rationale and predictions underlying this experiment.

_____	_____	_____
Participant's Printed Name	Participant's Signature	Date

_____	_____
Experimenter Name	Participant Number

Appendix B – Bi-digit Cursor Mapping Experiment Questionnaires

Finger Mapping Experiment – Pre-Questionnaire

Participant #: _____

Gender (circle one): *Male* *Female*

Age: _____

Handedness (circle one): *Left* *Right*

Which of the following single-point/single-finger touch-sensitive input devices have you used in the past (check all that apply)?

☐ *Laptop Touchpad*

☐ *Tablet with Pen/Stylus*

☐ *Tablet PC*

☐ *Touch Screen*

☐ *None*

Other (please specify): _____

Which of the following multi-point/multi-finger input devices have you used in the past (check all that apply)?

☐ *Powerbook Touchpad*

☐ *DiamondTouch*

☐ *SmartSkin*

☐ *SmartBoard*

☐ *Fingerworks*

☐ *None*

Other (please specify): _____

Finger Mapping Experiment – Post-Questionnaire

On a scale from 1 to 7, please rate the different cursor control techniques in terms of the **overall accuracy at which you could select targets** (where 1 is inaccurate and 7 is very accurate). **Please place an X at your desired location on the scale (between numbers is okay)**:

1 2 3 4 5 6 7 |----------|----------|----------|----------|----------|----------|----------| (inaccurate) (very accurate) *Index Finger Cursor*
1 2 3 4 5 6 7 |----------|----------|----------|----------|----------|----------|----------| (inaccurate) (very accurate) *Mid-point Cursor*
1 2 3 4 5 6 7 |----------|----------|----------|----------|----------|----------|----------| (inaccurate) (very accurate) *Thumb Cursor*

On a scale from 1 to 7, please rate the different cursor control techniques in terms of the **overall speed at which you could select targets** (where 1 means it took a very long time to select targets and 7 means targets could be selected very quickly). **Please place an X at your desired location on the scale (between numbers is okay)**:

1 2 3 4 5 6 7 |----------|----------|----------|----------|----------|----------|----------| (very slow) (very fast) *Index Finger Cursor*
1 2 3 4 5 6 7 |----------|----------|----------|----------|----------|----------|----------| (very slow) (very fast) *Mid-point Cursor*
1 2 3 4 5 6 7 |----------|----------|----------|----------|----------|----------|----------| (very slow) (very fast) *Thumb Cursor*

On a scale from 1 to 7, please rate the different cursor control techniques in terms of your **overall comfort when using them for selecting targets** (where 1 means you were very uncomfortable, and 7 means you were very comfortable). **Please place an X at your desired location on the scale (between numbers is okay)**:

1 2 3 4 5 6 7 |----------|----------|----------|----------|----------|----------|----------| (very uncomfortable) (very comfortable) *Index Finger Cursor*
1 2 3 4 5 6 7 |----------|----------|----------|----------|----------|----------|----------| (very uncomfortable) (very comfortable) *Mid-point Cursor*
1 2 3 4 5 6 7 |----------|----------|----------|----------|----------|----------|----------| (very uncomfortable) (very comfortable) *Thumb Cursor*

Please add any additional comments regarding the targeting experiment here:

_____

_____

_____

_____

_____

_____

_____

Appendix C – Finger Span Experiment Questionnaires

Finger Span Experiment – Pre-Questionnaire

Participant #: _____

Gender (circle one): *Male* *Female*

Age: _____

Handedness (circle one): *Left* *Right*

Which of the following single-point/single-finger input devices have you used in the past (check all that apply)?

☐ *Laptop Touchpad*

☐ *Tablet with Pen/Stylus*

☐ *Tablet PC*

☐ *Touch Screen*

☐ *None*

Other (please specify): _____

Which of the following multi-point/multi-finger input devices have you used in the past (check all that apply)?

☐ *Powerbook Touchpad*

☐ *DiamondTouch*

☐ *SmartSkin*

☐ *SmartBoard*

☐ *Fingerworks*

☐ *None*

Other (please specify): _____

Finger Span Experiment – Post-Questionnaire

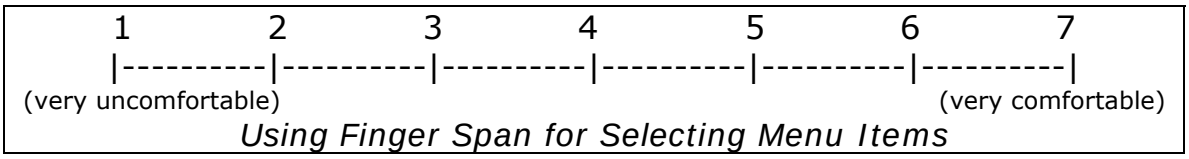
On a scale from 1 to 7, please rate the different menu visualizations in terms of the **overall accuracy at which you could select targets** (where 1 is inaccurate and 7 is very accurate). **Please place an X at your desired location on the scale (between numbers is okay)**:

1 2 3 4 5 6 7 |----------|----------|----------|----------|----------|----------|----------| (inaccurate) (very accurate) *Full Menu with All Options Visible*
1 2 3 4 5 6 7 |----------|----------|----------|----------|----------|----------|----------| (inaccurate) (very accurate) *Partial Menu with Only Single Option Visible*
1 2 3 4 5 6 7 |----------|----------|----------|----------|----------|----------|----------| (inaccurate) (very accurate) *Invisible Menu*

On a scale from 1 to 7, please rate the different menu visualizations in terms of the **overall speed at which you could select targets** (where 1 means it took a very long time to select targets and 7 means targets could be selected very quickly). **Please place an X at your desired location on the scale (between numbers is okay)**:

1 2 3 4 5 6 7 |----------|----------|----------|----------|----------|----------|----------| (very slow) (very fast) *Full Menu with All Options Visible*
1 2 3 4 5 6 7 |----------|----------|----------|----------|----------|----------|----------| (very slow) (very fast) *Partial Menu with Only Single Option Visible*
1 2 3 4 5 6 7 |----------|----------|----------|----------|----------|----------|----------| (very slow) (very fast) *Invisible Menu*

On a scale from 1 to 7, please rate the use of finger span in terms of your **overall comfort when using it to select menu targets** (where 1 means you were very uncomfortable, and 7 means you were very comfortable). **Please place an X at your desired location on the scale (between numbers is okay)**:



Using Finger Span for Selecting Menu Items

Please add any additional comments regarding the targeting experiment here:

_____

_____

_____

_____

_____

_____

_____

Bibliography

[Appl05] Apple. (2005). Powerbook G4 Touchpad. Retrieved on December 1, 2005, from: http://www.apple.com/powerbook

[Bala97] Balakrishnan, R., MacKenzie, I.S. (1997). Performance Differences in the Fingers, Wrist, and Forearm in Computer Input Control. In *Proceedings of ACM CHI*. p. 303-310.

[Bala98] Balakrishnan, R., Patel, P. (1998). The PadMouse: Facilitating Selection and Spatial Positioning for the Non-dominant Hand. In *Proceedings of ACM CHI*. p. 9-16.

[Baud93] Baudel, T., Beaudouin-Lafon, M. (1993). Charade: Remote Control of Objects Using Free-hand Gestures. In *Communications of the ACM*, 36(7). p. 28-35.

[Baud03] Baudisch, P., Cutrell, E., Robbins, D., Czerwinski, M., Tandler, P. Bederson, B., and Zierlinger, A. (2003). Drag-and-Pop and Drag-and-Pick: Techniques for Accessing Remote Screen Content on Touch- and Pen-operated Systems**.** In *Proceedings of Interact.* p. 57-64.

[Benk06] Benko, H., Wilson, A., Baudisch, P. (2006). Precise Selection Techniques for Multi-touch Screens. In *Proceedings of ACM CHI*. p. 1263-1272.

[Beze05] Bezerianos, A and Balakrishnan, R. (2005). The Vacuum: Facilitating the Manipulation of Distant Objects. In *Proceedings of ACM CHI*. p. 361-370.

[Bier93] Bier, E., Stone, M., Pier, K., Buxton, B., DeRose, T. (1993). Toolglass and Magic Lenses: The See-Through Interface. In *Proceedings of ACM SIGGRAPH*. p. 73-80.

[Blas04] Blasko, G., Feiner, S. (2004). Single-handed Interaction Techniques for Multiple Pressure-Sensitive Strips. In *Proceedings of ACM CHI (Late Breaking Results)*. p. 1461-1464.

[Bove00] Van Boven, R., Hamilton, R., Kauffman, T., Keenan, J., Pascual-Leone. A. (2000). Tactile Spatial Resolution in Blind Braille Readers. In *Neurology*, 54(12). p. 2230-2236.

[Buch04] Buchmann, V., Violich, S., Billinghurst, M., Cockburn, A. (2004). FingARtips – Gesture Based Direct Manipulation in Augmented Reality. In *Proceedings of Graphite 2004*. p. 212-221.

[Buxt85] Buxton, W., Hill, R., Rowley, P. (1985). Issues and techniques in touch-sensitive tablet input. In *Proceedings of ACM SIGGRAPH*. p. 215-223.

[Buxt90] Buxton, W. (1990). A Three-State Model of Graphical Input. In D. Diaper et al. (Eds), *Human-Computer Interaction – Interact '90*. Amsterdam: Elsevier Science Publishers. p. 449-456.

[Buxt92] Buxton, W. (1992). Telepresence: integrating shared task and person spaces. In *Proceedings of Graphics Interface*. p. 123-129.

[Cao03] Cao, X., and Balakrishnan, R. (2003). VisionWand: Interaction Techniques for Large Displays Using a Passive Wand Tracked in 3D. In *Proceedings of ACM UIST*. p. 173-182.

[Card91] Card, S., MacKinlay, J., Robertson, G. (1991). A Morphological Analysis of the Design Space of Input Devices. In *ACM Transactions on Computer-Human Interaction*, Vol. 9, No. 2, April 1991. p. 99-122.

[Cham03] Cham, T-J., Rehg, J., Sukthankar, R., Sukthankar, G. (2003). Shadow Elimination and Occluder Light Suppression for Multi-Projector Displays. In *Proceedings of CVPR*. p. 513-520.

[Cors03] Corso, J., Burschka, D., Hager, D. (2003). The 4DT: Unencumbered HCI with VICs. In *Proceedings of IEEE Workshop on Computer Vision for Human Computer Interaction (CVPR-HCI)*.

[Cutl97] Cutler, L., Frohlich, B., Hanrahan, P. (1997). Two-Handed Direct Manipulation on the Responsive Workbench. In *Proceedings of ACM I3D*. p. 107-114.

[Davi02] Davis, J., Chen, X. (2002). Lumipoint: Multi-User Laser-Based Interaction on Large Tiled Displays. In *Displays*, Volume 23, Issue 5.

[Diet01] Dietz, P., Leigh, D. (2001). DiamondTouch: A Multi-user Touch Technology. In *Proceedings of ACM UIST*. p. 219-226.

[Ekma67] Ekman, G., Berglund, B., Berglund, U., Lindvall, T. (1967). Perceived intensity of odor as a function of time of adaptation. In *Scandinavian Journal of Psychology*, 8. p. 177-186.

[Etie99] Etievant, P., Callement, G., Langlois, D., Issanchou, S., Coquibus, N. (1999). Odor Intensity Evaluation in Gas Chromatography-Olfactometry by Finger Span Method. In *Journal of Agricultural and Food Chemistry*, 47(4). p. 1673-1680.

[Fake05] Fakespace Systems, Inc. (2005). The Pinch Glove. Retrieved on December 1, 2005, from: http://www.fakespace.com/pinch.htm

[Faug01] Faugeras, O., Luong, Q. (2001). *The Geometry of Multiple Images*. The MIT Press.

[Fial04] Fiala, M. (2004). ARTag Revision 1, A Fiducial Marker System Using Digital Techniques. NRC/ERB-1117 Technical Report. National Research Council of Canada.

[Fing05] FingerWorks, Inc. (2005). TouchStream LP Input Device. Retrieved on December 1, 2005, from: http://www.fingerworks.com

[Fitt54] Fitts, P. (1954). The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement. In *Journal of Experimental Psychology*, 47(6). p. 381-391.

[Geib98] Geißler, J. (1998). Shuffle, Throw or Take It! Working Efficiently with an Interactive Wall. (1998). In *ACM CHI, Extended Abstracts*. p. 265-266.

[Gest05] GestureTek, Inc. (2005). GestPoint Hand Tracking Technology. Retrieved on December 2, 2005, from http://www.gesturetek.com

[Grah96] Graham, E., MacKenzie, C. (1996). Physical Versus Virtual Pointing. In *Proceedings of ACM CHI*. p. 292-299.

[Gros04] Grossman, T., Wigdor, D., Balakrishnan, R. (2004). Multi-finger Gestural Interaction with 3D Volumetric Displays. In *Proceedings of ACM UIST*. p. 61-70.

[Guia87] Guiard, Y. (1987). Asymmetric Division of Labor in Human Skilled Bimanual Action: The Kinematic Chain as a Model. In *Journal of Motor Behavior, 1987, 19*. p. 486-517.

[Guim00] Guimbretière, F., Winograd, T. (2000). FlowMenu: Combining Command, Text, and Data Entry. In *Proceedings of ACM UIST*. p. 213-216.

[Guim01] Guimbretière, F., Stone, M., Winograd, T. (2001). Fluid Interaction with High-resolution Wall-size Displays. In *Proceedings of ACM UIST*. p. 21-30.

[Guim05] Guimbretière, F., Martin, A., Winograd, T. (2005). Benefits of Merging Command Selection and Direct Manipulation. In *ACM Transactions on Computer-Human Interaction (TOCHI)*, 12(3). p. 460-476.

[Hage00] Hager-Ross, C., Schieber, M. (2000). Quantifying the Independence of Human Finger Movements: Comparisons of Digits, Hands, and Movement Frequencies. In *The Journal of Neuroscience*, 20(22):8542-8550.

[Han05] Han, J. (2005). Low-cost Multi-touch Sensing Through Frustrated Total Internal Reflection. In *Proceedings of ACM UIST*. p. 115-118.

[Hard01] Hardenberg, C. V., Bérard, F. (2001). Bare-Hand Human-Computer Interaction. In *Proceedings of the Workshop on Perceptive User Interface*.

[Himb03] Himberg, J., Hakkila, J., Kangas, P., Mantyjarvi, J. (2003). On-line Personalization of a Touch Screen Based Keyboard. In *Proceedings of Intelligent User Interfaces (IUI)*. p. 77-84.

[Hinc97] Hinckley, K., Pausch, R., Proffitt, D., Patten, J., Kassell, N. (1997). Cooperative Bimanual Action. In *Proceedings of ACM CHI*. p. 27-34.

[Hinc99] Hinckley, K., Sinclair, M. (1999). Touch-sensing Input Devices. In *Proceedings of ACM CHI*. p. 223-230.

[Hump02] Humphreys, G., Houston, M., Ng, R., Frank, R., et al. (2002). Chromium: A Stream-Processing Framework for Interactive Rendering on Clusters. In *ACM Transactions on Graphics, Proceedings of ACM SIGGRAPH*. p. 693-702.

[Igar05] Igarashi, T., Moscovich, T., Hughes, J. (2005). As-Rigid-As-Possible Shape Manipulation. In *ACM Transactions on Graphics, Proceedings of ACM SIGGRAPH*. p. 157-164.

[Imme05] Immersion Corp. (2005). CyberGlove. Retrieved on December 1, 2005, from: http://www.immersion.com/3d/products/cyber_glove.php

[Ishi92] Ishii, H., Kobayashi, M. (1992). ClearBoard: a seamless medium for shared drawing and conversation with eye contact. In *Proceedings of ACM CHI*. p. 525-532

[Izad03] Izadi, S., Brignull, H., Rodden, T., Rogers, Y., Underwood, M. (2003). Dynamo: A Public Interactive Surface Supporting the Cooperative Sharing and Exchange of Media. In *Proceedings of ACM UIST*. p. 159-168.

[Jast86] Jastrow, J. (1886). The perception of space by disparate senses. In *Mind*, 11(44): 539-554.

[Joha02a] Johanson, B., Fox, A., and Winograd, T. (2002). The Interactive Workspace Project: Experiences with Ubiquitous Computing Rooms. In *IEEE Pervasive Computing*. p. 67-74.

[Joha02b] Johanson, B., Hutchins, G., Winograd, T., Stone, M. (2002). Pointright: Experience with Flexible Input Redirection in Interactive Workspaces. In *Proceedings of ACM UIST*. p. 227-234.

[Kabb94] Kabbash, P., Buxton, W., Sellen, A. (1994). Two-Handed Input in a Compound Task. In *Proceedings of ACM CHI*. p. 417-423.

[Kara05] Karam, M., Schraefel, M. (2005). A Taxonomy of Gestures in Human Computer Interaction. Technical Report ECSTR-IAM05-009, Electronics and Computer Science, University of Southampton.

[Khan04] Khan, A., Fitzmaurice, G., Almeida, D., Burtnyk, N., Kurtenbach, G. (2004). A Remote Control Interface for Large Displays. In *Proceedings of ACM UIST*. p. 127-136.

[Khan05] Khan, A., Matejka, J., Fitzmaurice, G., Kurtenbach, G. (2005). Spotlight: Directing Users' Visual Attention on Large Displays. In *Proceedings of the ACM CHI*. p. 791-798.

[Kirs94] Kirsh, D., & Maglio, P. (1994). On Distinguishing Epistemic from Pragmatic Action. In *Cognitive Science,* 18. p. 513-549.

[Kirs98] Kirstein, C. and Muller, H. (1998). Interaction with a Projection Screen using a Camera-tracked Laser Pointer. In *Proceedings of Multimedia Modeling*. p. 191.

[Kjel97] Kjeldsen, F. (1997). Visual Interpretation of Hand Gestures as a Practical Interface Modality. *PhD Thesis*, Columbia University.

[Kjel01] Kjeldsen, R., and Hartman, J. (2001). Design Issues for Vision-based Computer Interaction Systems. In *Proceedings of the Workshop on Perceptive User Interfaces*.

[Kols02] Kolsch, M., Turk, M. (2002). Keyboards without Keyboards: A Survey of Virtual Keyboards. Technical Report 2002-21. University of California, Santa Barbara.

[Krue85] Krueger, M. (1985). Videoplace – An Artificial Reality. In *Proceedings of ACM CHI*. p. 35-40.

[Kurt93] Kurtenbach, G., Buxton, W. (1993). The Limits of Expert Performance Using Hierarchic Marking Menus. In *Proceedings of ACM CHI*. p. 482-487.

[Kurt97] Kurtenbach, G., Fitzmaurice, G., Baudel, T., Buxton, B. (1997). The Design of a GUI Paradigm based on Tablets, Two-hands, and Transparency. In *Proceedings of ACM CHI*. p. 35-42.

[Lavi99] LaViola, J., Zeleznik, R. (1999). Flex and Pinch: A Case Study of Whole Hand Input for Virtual Environment Interaction. In *Proceedings of IASTED International Conference on Computer Graphics and Imaging*. p. 221-225.

[Lede03] Lederman, S., Wing, A. (2003). Perceptual Judgement, Grasp Point Selection, and Object Symmetry. In *Experimental Brain Research*, 152(2). p. 156-165.

[Lee85] Lee, SK., Buxton, W., Smith, K. (1985). A Multi-Touch Three Dimensional Touch-Sensitive Tablet. In *Proceedings of ACM CHI*. p. 21-25.

[Loom83] Loomis, J., Poizner, H., Bellugi, U., Blakemore, A., Hollerbach, J. (1983). Computer Graphic Modeling of American Sign Language. In *International Conference on Computer Graphics and Interactive Techniques*. p. 105-114.

[Mack94] MacKenzie, C., Iberall, T. (1994). *The Grasping Hand*. Amsterdam: North Holland, Elsevier Science.

[Mali04] Malik, S., Laszlo, J. (2004). The Visual Touchpad: A Two-Handed Gestural Input Device. In *Proceedings of ACM ICMI*. p. 289-296.

[Mali05] Malik, S., Ranjan, A., Balakrishnan, R. (2005). Interacting with Large Displays from a Distance with Vision-Tracked Multi-Finger Gestural Input. In *Proceedings of ACM UIST*. p. 43-52.

[Mali06] Malik, S., Singh, K. (2006). Interactive Art Installation. Deaf Culture Centre, Distillery Historic District, 55 Mill Street, Suite 101, Toronto, Ontario, Canada. http://www.deafculturecentre.com

[Mali07a] Malik, S., Balakrishnan, R., Jepson, A. (2007). Bi-digit Widgets: Using the Thumb and Index Finger Asymmetrically on Touch-sensitive Surfaces. *In submission*.

[Mali07b] Malik, S., Balakrishnan, R., Jepson, A. (2007). An Evaluation of Finger Span Perception for Bi-digital Input. *In submission*.

[Masl00] Masliah, M., Milgram, P. (2000). Measuring the Allocation of Control in a 6 Degree-of-Freedom Docking Experiment. In *Proceedings of ACM CHI*. p. 25-32.

[Mats97] Matsushita, N., Rekimoto, J. (1997). HoloWall: Designing a Finger, Hand, Body, and Object Sensitive Wall. In *Proceedings of ACM UIST*. p. 209-210.

[Mats00] Matsushita, N., Ayatsuka, Y., Rekimoto, J. (2000). Dual Touch: A Two-handed Interface for Pen-based PDAs. In *Proceedings of ACM UIST*. p. 211-212.

[Micr05] Microsoft. (2005). Pointer Ballistics for Windows XP. Retrieved on December 1, 2005, from: http://www.microsoft.com/whdc/device/input/pointer-bal.mspx

[Mosc06] Moscovich, T., Hughes, J. (2006). Multi-finger Cursor Techniques. In *Proceedings of Graphics Interface*. To appear.

[Mosc07] Moscovich, T. (2007). Principles and Applications of Multi-touch Interaction. *PhD Thesis*, Brown University.

[Mott01] Mottet, D., Guiard, Y., Ferrand, T., Bootsma, R. (2001). Two-handed Performance of a Rhythmical Fitts Task by Individuals and Dyads. In *Journal of Experimental Psychology: Human Perception and Performance*, 27(6). p. 1275-1286.

[Myna99] Mynatt, E., Igarashi, T., Edwards, W., LaMarca, A. (1999). Flatland: New Dimensions in Office Whiteboards. In *Proceedings of ACM CHI*. p. 346-353.

[Oka02] Oka, K., Sato, Y., Koike, K. (2002). Real-time Fingertip Tracking and Gesture Recognition. In *Proceedings of IEEE Computer Graphics and Applications*, 22(6). p. 64-71.

[Pede93] Pederson, E., McCall, K., Moran, T., Halasz, F. (1993). Tivoli: An Electronic Whiteboard for Informal Workgroup Meetings. In *Proceedings of ACM CHI*. p. 391-398.

[Perl98] Perlin, K. (1998). Quikwriting: Continuous Stylus-based Text Entry. In *Proceedings of ACM UIST*. p. 215-216.

[Pott88] Potter, R., Weldon, L., Shneiderman, B. (1988). Improving the Accuracy of Touch Screens: An Evaluation of Three Strategies. In *Proceedings of ACM CHI*. p. 27-32.

[Raj99] Raj, R., Marquis, C. (1999). Finger Dominance. In *The Journal of Hand Surgery (British and European Volume)*, 24B(4). p. 429-430.

[Ramo04] Ramos, G., Boulos, M., Balakrishnan, R. (2004). Pressure Widgets. In *Proceedings of ACM CHI*. p. 487-494.

[Reki02] Rekimoto, J. (2002). SmartSkin: An Infrastructure for Freehand Manipulation on Interactive Surfaces. In *Proceedings of ACM CHI*. p. 213-220.

[Reil04] Reilly, K., Hammond, G. (2004). Human Handedness: Is there a Difference in the Independence of the Digits on the Preferred and Non-preferred Hands. In *Experimental Brain Research*, 156(2), p. 255-262.

[Rime91] Rime, B., Schiaratura, L. (1991). Gesture and Speech. In *Fundamentals of Nonverbal Behavior*. R. Feldman, B. Rime (Eds). Cambridge University Press.

[Ring01] Ringel, M., Berg, H., Jin, Y., Winograd, T. (2001). Barehands: Implement-free Interaction with a Wall-mounted Display. In *Proceedings of ACM CHI Extended Abstracts*. p. 367-368.

[Rous01] Roussel, N. (2001). Exploring new uses of video with videoSpace. In *Proceedings of the IFIP Conference on Engineering for HCI*, Volume 2254 of Lecture Notes in Computer Science, Springer. p. 73-90.

[Sant97] Santello, M., Soechting, J. (1997). Matching Object Size by Controlling Finger Span and Hand Shape. In *Somatosensory & Motor Research*, 14(3). p. 203-212.

[Sant98] Santello, M., Flanders, M., Soeghting, J. (1998). Postural Hand Synergies for Tool Use. In *Journal of Neuroscience*, 18(23). p. 10105-10115.

[Sege98] Segen, J., Kumar, S. (1998). GestureVR: Vision-based 3D Hand Interface for Spatial Interaction. In *Proceedings of ACM Multimedia*. p. 455-464.

[Shne83] Shneiderman, B. (1983). Direct Manipulation: A Step Beyond Programming Languages. In *IEEE Computer*, 16(8). p. 57-69.

[Shne91] Shneiderman, B. (1991). Touch Screens Now Offer Compelling Uses. In *IEEE Software*, Vol.8, No.2, p. 93-94, 107.

[Smar05] Smart Technologies. (2005). The DViT SMARTBoard. Retrieved on December 1, 2005, from: http://www.smarttech.com/DViT

[Stot03] Stotts, D., Smith, J., and Jen, D. (2003). The Vis-a-Vid Transparent Video FaceTop. In *Proceedings of ACM UIST*. p. 57-58.

[Stur89] Sturman, D., Zeltzer, D., Pieper, S. (1989). Hands-on Interaction with Virtual Environments. In *Proceedings of ACM UIST*. p. 19-24.

[Stur92] Sturman, D. (1992). Whole-hand Input. *PhD Thesis*, Massachusetts Institute of Technology.

[Stur94] Sturman, D., Zeltzer, D. (1994). A Survey of Glove-based Input. In *IEEE Computer Graphics and Applications*, 14(1). p. 30-39.

[Sue02] Sue, L. (2002). Piano Techniques - Finger Independence I. Retrieved on December 20, 2006 from: http://choraegus.com/LW/dissertations/pianoTechnique/fingerindependence1.html

[Sugi98] Sugiura, A., Koseki, Y. (1998). A User Interface Using Fingerprint Recognition: Holding Commands and Data Objects on Fingers. In *Proceedings of ACM UIST*. p. 71-79.

[Swam97] Swaminathan, K, and Sato, S. (1997). Interaction Design for Large Displays. *Interactions,* Volume 4, Issue 1.

[Syna05] Synaptics, Inc. (2005). Synaptics Touchpad. Retrieved on December 2, 2005, from: http://www.synaptics.com

[Tact05] Tactiva. (2005). TactaPad Input Device. Retrieved on December 2, 2005, from: http://www.tactiva.com

[Tang91] Tang, J., Minneman, S. (1991). Videowhiteboard: video shadows to support remote collaboration. In *Proceedings of ACM CHI*. p. 315-322.

[Truc98] Trucco, E., Verri, A. (1998). *Introductory Techniques for 3-D Computer Vision*. Prentice-Hall.

[Vand95] van Doren, C. (1995). Cross-modality Matches of Finger Span and Line Length. In *Perception & Psychophysics*, 57(4). p. 555-568.

[Voge04] Vogel, D., Balakrishnan, R. (2004). Interactive public ambient displays: transitioning from implicit to explicit, public to personal, interaction with multiple users. In *Proceedings of ACM UIST*. p. 137-146.

[Voge05] Vogel, D., Balakrishnan, R. (2005). Distant Freehand Pointing and Clicking on Very Large, High Resolution Displays. In *Proceedings of ACM UIST*, 2005. p. 33-42.

[Wang00] Wang, Y., MacKenzie, C. (2000). The Role of Contextual Haptic and Visual Constraints on Object Manipulation in Virtual Environments. In *Proceedings of ACM CHI*. p. 532-539.

[Well93] Wellner, P. (1993). Interacting with Paper on the Digital Desk. In *Communications of the ACM*, Vol. 26, No. 7, July 1993. p. 87-96.

[Wils04] Wilson, A. (2004). TouchLight: An Imaging Touch Screen and Display for Gesture-based Interaction. In *Proceedings of ACM ICMI*. p. 69-76.

[Wils05] Wilson, A. (2005). PlayAnywhere: A Compact Interactive Tabletop Projection-vision System. In *Proceedings of ACM UIST*. p. 83-92.

[Wu03] Wu, M., Balakrishnan, R. (2003). Multi-finger and Whole Hand Gestural Interaction Techniques for Multi-user Tabletop Displays. In *Proceedings of ACM UIST*. p. 193-202.

[Yee04] Yee, K-P. (2004). Two-handed Interaction on a Tablet Display. In *Proceedings of ACM CHI (Short Papers)*. p. 1493-1496.

[Zhai96] Zhai, S., Milgram, P., Buxton, W. (1996). The Influence of Muscle Groups on Performance of Multiple Degree-of-Freedom Input. In *Proceedings of ACM CHI*. p. 308-315.

[Zhai98] Zhai, S., Milgram, P. (1998). Quantifying Coordination in Multiple DOF Movement and its Application to Evaluating 6 DOF Input Devices. In *Proceedings of ACM CHI*. p. 320-327.

[Zhan99] Zhang, Z. (1999). Flexible Camera Calibration by Viewing a Plane from Unknown Orientations. In *Proceedings of ICCV*. p. 666-673.

[Zhan01] Zhang, Z., Wu, Y., Shan, Y., Shafer, S. (2001). Visual Panel: Virtual Mouse, Keyboard, and 3D Controller with an Ordinary Piece of Paper. In *Proceedings of ACM Workshop on Perceptive User Interfaces*. p. 1-8.

[Zimm87] Zimmerman, T., Lanier, J., Blanchard, C., Bryson, S., Harvill, Y. (1987). A Hand Gesture Interface Device. In *Proceedings of ACM CHI*. p. 189-192.