Investigating a Multi-touch User Interface for Three-dimensional CAD Operations

A Thesis Presented

By

Srinivasan Radhakrishnan

To

Department of Mechanical and Industrial Engineering

in partial fulfillment of the requirements

for the degree of

Masters of Science

in

Computer System Engineering

Northeastern University
Boston, Massachusetts

May 2010
Thesis Title: Investigating a Multi-touch User Interface for Three-dimensional CAD Operations

Author: Srinivasan Radhakrishnan

Department: Mechanical and Industrial Engineering

Approved for Thesis requirement of the Master of Science Degree

Thesis Advisor, Dr. Ibrahim Zeid

Thesis Advisor, Dr. Sagar V. Kamarthi

Chairman of the MIE Department, Dr. Hameed Metghalchi

Director of Graduate School, Dean Yaman Yener
Abstract

The area of multitouch interaction research is at its infancy. The commercial sector has seen an exponential growth in this area with ubiquitous products like Apple i-Phone and Microsoft surface table. In spite of their popularity, developers are still finding it difficult to extend this novel interface to engineering applications such as Computer Aided Design (CAD), due to insufficient understanding of the factors affecting the multitouch interface when applied to CAD operations. This problem is magnified when it comes to 3D CAD operations. The aim of this research is to (1) study the key elements of the multitouch interface, (2) identify the factors affecting the performance of a multitouch-enabled 3D modeling environment, and (3) lay a foundation for future research and highlight the direction for researchers and developers to extend the multitouch interface for CAD modeling operations and other engineering applications.

We have conducted mouse emulation experiments in order to achieve the goal. We compare the standard mouse device with two finger-based touch interaction techniques (drag state finger touch and track state finger touch) for 3D CAD modeling operations. The mouse device outperformed the finger touch techniques and yielded statistically better performance results in terms of both task completion time and error rates. The paired t-test indicated that both the task completion time and error rates are statistically the same for the two finger touch techniques under investigation. However, the error concentration observed from the experiments revealed that the track state technique is more suited for edge selection tasks as compared to the drag state technique. Regardless, both the finger touch techniques suffered from lack of precise dimension control while executing the tasks. The presence of grid had a significant desirable effect on the error rate.
Acknowledgement

I would like to express my sincere gratitude to my thesis advisor Dr. Ibrahim Zeid for his tremendous support and guidance. Dr. Zeid’s profound knowledge in the field of CAD provided the right direction for this research. I would also like to thank Dr. Sagar V. Kamarthi for his continuous support and Dr. Yingzi Lin for her support and guidance which ensured proper environment for the experiments. I feel proud to be a part of Mechanical and Industrial engineering department at Northeastern which enabled me to increase my knowledge base.

I would like to thank my parents for their efforts to create a prosperous life and Yogita Pai for her tremendous support and critical review. I am very grateful to my friend Arjun who helped me in setting up the experiments.

Srinivasan Radhakrishnan
<table>
<thead>
<tr>
<th>Table of Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF TABLES ........................................................................1</td>
</tr>
<tr>
<td>LIST OF FIGURES ......................................................................2</td>
</tr>
<tr>
<td>CHAPTER 1 ..............................................................................5-7</td>
</tr>
<tr>
<td>INTRODUCTION ..........................................................................5</td>
</tr>
<tr>
<td>1.1 Overview ...........................................................................5</td>
</tr>
<tr>
<td>1.2 Research Objectives ............................................................6</td>
</tr>
<tr>
<td>1.3 Motivation ..........................................................................6</td>
</tr>
<tr>
<td>CHAPTER 2 ..............................................................................8-14</td>
</tr>
<tr>
<td>COMPARING GESTURE AND STANDARD INPUT DEVICES ............8</td>
</tr>
<tr>
<td>2.1 Properties of Input Devices .................................................9</td>
</tr>
<tr>
<td>2.1.1 Sensing Physical Properties .............................................9</td>
</tr>
<tr>
<td>2.1.2 Dimensionality ................................................................9</td>
</tr>
<tr>
<td>2.1.3 Input Device States ............................................................10</td>
</tr>
<tr>
<td>2.1.4 Direct and Indirect Control ..............................................10</td>
</tr>
<tr>
<td>2.1.5 Device Acquisition Time ..................................................11</td>
</tr>
<tr>
<td>2.2 Pointing Devices ...............................................................11</td>
</tr>
<tr>
<td>2.2.1 Mouse ..........................................................................11</td>
</tr>
<tr>
<td>2.2.2 Trackball .......................................................................11</td>
</tr>
<tr>
<td>2.2.3 Tablets ........................................................................12</td>
</tr>
</tbody>
</table>
2.2.4 Pen Devices .................................................................12
2.2.5 Joystick .................................................................12
2.2.6 Keyboard .................................................................13
2.2.7 Gesture Based Input ..................................................13
2.2.8 Multitouch Input ......................................................14

CHAPTER 3 .................................................................15-33

HARDWARE DESIGN OF MULTITOUCH INTERFACE ..................15

3.1 Classification of Multitouch Hardware ...................................15

3.1.1 Optical Based Touch Surfaces .........................................16

3.2 Choosing Components for Optical Based Multitouch Surfaces ......24

3.2.1 Camera .....................................................................24

3.2.2 DLP Projector .............................................................25

3.3 Optical Based Multitouch Image Processing Software ...............26

3.3.1 Blob Detection ..........................................................28

3.3.2 Blob Tracking ............................................................29

3.3.3 Generating Events .....................................................32

3.3.4 Communicating the Events ...........................................33

CHAPTER 4 .................................................................34-40

EVALUATION TECHNIQUES OF INPUT DEVICES ..................34

4.1 Fitts Law ......................................................................34

4.2 Fitts Law for Two Dimensional Tasks ....................................36

4.3 Steering Law ..................................................................38
CHAPTER 5

KEY ELEMENTS OF THE MULTITOUCH INTERFACE

5.1 Multitouch Interface Research

5.2 Multitouch Software Research

5.2.1 Touch Input Properties

5.2.2 GUI Elements Design

5.2.3 GUI Elements Layout

5.2.4 Gestures

5.3 Multitouch Hardware Research

5.4 Computer Aided Design

5.4.1 Selection Task

5.4.2 Dimension Task

5.4.3 Gestures

CHAPTER 6

EVALUATION OF MULTITOUCH INTERFACE

6.1 Unimanual Experiment

6.2 Unimanual Experiment Results

6.3 Bimanual Experiment

CHAPTER 7

DESIGN AND EVALUATION OF 3D CAD FLOATING TOOLBAR

7.1 Experiment Setup

7.2 Experimental Procedure
7.3 Experimental Results .................................................................66
7.4 Discussion .................................................................................71
7.5 Proposed Floating Toolbar for 3D CAD Operations .................75
7.6 Proposed Evaluation Technique for Gesture based CAD Operations........77
7.7 Conclusion and Future Work.........................................................80

REFERENCES ..................................................................................82-89
LIST OF TABLES

Table 1: Comparison of optical based multitouch system……………………………….. 23
Table 2: Broad comparison of the different optical based multitouch system…………… 23
Table 3: Transition matrix indicating all possible states………………………………….. 31
LIST OF FIGURES

Figure 1: Optical based multitouch tracking pipeline .................................................. 17

Figure 2: FTIR setup without Compliant layer and the corresponding finger blob contrast... 19

Figure 3: Alternate FTIR setup with Compliant layer ..................................................... 19

Figure 4: Rear side illumination setup with corresponding finger blob contrast .............. 21

Figure 5: Front side illumination setup ........................................................................... 21

Figure 6: Diffused surface illumination setup with corresponding finger blob contrast .... 22

Figure 7: Throw ratio of a projector ................................................................................ 26

Figure 8: CCV images of a FTIR setup ........................................................................... 27

Figure 9: CCV images of a RI setup ............................................................................... 28

Figure 10: CCV images of Blob Detection ..................................................................... 29

Figure 11: Blobs detected in the previous frame and current frame ............................... 30

Figure 12: Final results of the blob tracking process ..................................................... 32

Figure 13: Schematic view of the OSC communication pipeline .................................. 33

Figure 14: Fitts reciprocal tapping device ...................................................................... 34

Figure 15: Determining effective target width ............................................................... 37

Figure 16: Two goal passing experiment ....................................................................... 39
Figure 17: Steering through a tunnel…………………………………………………………..39

Figure 18: General steering law………………………………………………………………40

Figure 19: Schematic diagram showing the key elements of a multitouch interface…………..41

Figure 20: Physical finger properties and event property, “Zoom gesture” …………………..43

Figure 21: Examples of direct gesture and symbolic gesture…………………………………47

Figure 22: Illustration of various surface and solid modeling operations in CAD…………….50

Figure 23: Selection of a point using mouse pointer and finger touch …………………..51

Figure 24: Selection of an intersecting point using mouse pointer and finger touch ………….52

Figure 25: Gesture based modifications………………………………………………………54

Figure 26: Unimanual experiment………………………………………………………………56

Figure 27: Finger touch variation according to table orientation; vertical and horizontal…….57

Figure 28: Bimanual experiment……………………………………………………………….58

Figure 29: TSFT technique…………………………………………………………………….62

Figure 30: FTIR based setup used for the touch experiments………………………………….63

Figure 31: Sub tasks associated with the experiment………………………………………….64

Figure 32: Average completion time and error rate according to toolbar position…………….66

Figure 33: Error concentration for different subtasks……………………………………….73
**Figure 34:** Proposed floating toolbar for 3D CAD operations........................................76

**Figure 35:** Schematic diagram of an operation cycle and sub-task .................................77
Chapter 1

Introduction

1.1 Overview

The Multitouch interface uses a natural method of human interaction, mainly the finger touch as the primary method of input. The users can use single or multiple fingers to touch the screen in order to trigger an event. Moreover the developers can take the advantage of multiple finger touch to form a gesture which in turn can be used for performing various computer related tasks. The research in the area of multitouch interaction has been around for a long time. Since the introduction of FTIR based multitouch table by Jeff Han in 2005 [30] and commercial adoption of multitouch systems in products like apple i-phone and HP-slate; similarly the hardware and software supporting multitouch interface have grown and evolved exponentially. Multitouch interface has attracted many people to the world of finger gestures. Two finger gestures to scale and rotate images are a common sight today. For example, i-phone applications enable users to use natural finger gestures for tasks such as playing games, browsing, socializing and so on. Envisioning multitouch as the future of Human Computer interaction, one can see many research opportunities to address issues in multitouch hardware and software domain. At present the most common computer input device is the mouse and keyboard. The software interface relies heavily on GUI elements like button, menu, text field, etc. The software systems are mainly categorized as (1) application software, (2) system software, (3) computer programming tools. Application software are tools for performing general tasks such as word editing and web browsing and specific tasks such as accounting and photo editing. System software, a generic term, refers to
the computer programs used to start and run computer systems and networks. Computer programming tools include compilers and linkers, used to translate and combine computer program source code and libraries into executable programs.

In spite of the popularity of the multitouch interface for application software, there is a dearth of knowledge when it comes to deciding the types of application that are suitable for multitouch interaction. Engineering software applications like Computer Aided Design (CAD) for modeling and animation are still in a dilemma to incorporate multitouch interaction for their application.

1.2 Research Objectives

The main goal of this research is to understand the dynamics of using multitouch interface for 3D modeling tasks. The objectives of the thesis are stated as follows:

- Identify and outline the components of a multitouch interface
- Identify the key elements for building a multitouch software interface
- Compare the performance of 3D CAD operations using multitouch and mouse interface
- Investigating the parameters that effect the performance of the task
- Outline the future research direction for introducing multitouch interfaces for 3D modeling environment

1.3 Motivation

The research community has explored novel interaction techniques for 2D CAD. The area of combining multitouch interaction and 3D CAD modeling is least explored. The pros and cons of using this novel interaction technique for CAD modeling operations are not well understood. Taking a decision to incorporate multitouch system in current applications depends on a very good multitouch interface design which can be achieved only when we have deep knowledge about the parameters that affect the interface design pertaining to CAD modeling environment.
Hence this study will lay the foundation for academic research and engineering applications to incorporate the multitouch system in their existing applications.
Chapter 2

Comparing Gesture and Standard Input Devices

The Association for Computing Machinery defines human-computer interaction (HCI) as "a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them". HCI uses knowledge field from both human and machine side. On the machine side, techniques in computer graphics, operating systems, programming languages, and development environments are relevant. On the human side, communication theory, graphic and industrial design disciplines, linguistics, social sciences, cognitive psychology, and human factors are relevant. According to Hinckley, et al. [1], an interaction technique is a fusion of input and output, consisting of all the hardware and software elements, that provides a way for users to accomplish low level tasks. It is interesting to observe the evolution of input and output devices. The evolution begins from clear difference in input and output devices in the past to complete integration of these devices at present and in future. This evolution is evident from clearly differentiated mouse and monitor interface to multitouch system which integrates both input and output interface. The research for total integration of input and output devices have given birth to systems like ubiquitous computing [2] and tangible interaction [3]. An input device for computer is a system that senses information from the physical domain and sends them to the computer. An output device reflects the change in environment through devices such as a monitor or display unit. Apart from these devices there are feedback systems such as speakers, tactile devices and force feedback (Haptic) devices. The human part in HCI performs some tasks to achieve a specific goal. Some of the tasks are repetitive and hence termed as elemental tasks.
Chan, et al. [4] identified these elemental tasks as, selection, position, orient, path, quantify and text. Hinckley, et al. [1], argued that the six elemental tasks are not complete since this individually or collectively fails to accommodate devices such fingerprint scanner, which can be used for password replacement.

2.1 Properties of Input Devices

The general properties of inputs are discussed below.

2.1.1 Sensing Physical Properties

There are different physical properties sensed by different input devices. A mouse senses motion; a tablet senses position while an isometric joystick (included in IBM laptops) senses force. A rotary device senses change in angle and torque. Based on the type of properties sensed, pointing devices are classified as absolute input device whereas motion devices are classified as relative input devices. Hinckley, et al. [1], states that the key difference between an absolute input device and relative input device is that while the former can support both absolute motion and relative motion (change in position relative to the previous motion) the later can only support relative motion. However, a relative input device can only emulate absolute position by introducing a cursor on the screen. Another key difference known as “muscle memory” [5] highlights the difficulty of moving a mouse cursor to an area (except edges) without looking at screen, but the same task is achievable with tablet and stylus using kinesthetic sense.

2.1.2 Dimensionality

The input devices can have linear DOF or angular DOF or both. A mouse can measure along two linear dimensions while a rotary knob has one angular DOF. There are devices with 6 DOF which can measure three linear and three angular dimensions. As per Hinckley, et al. [1], if the number of dimensions required for performing the task does not match the number of dimensions
provided by the input device then the interaction may require, buttons or widgets mode switching. This creates a problem for three dimensional user interfaces [6] [7]. Several techniques have been proposed to allow existing 2D devices to control 3D positioning and orientation [8] [9] [10]. Hinckley, et al. [1] informed that, though multiple DOF can offer superior performance [11] [12], it may be ineffective for standard desktop tasks and hence more attention should be given to overall performance [13] [14].

2.1.3 Input Device States

Bill Buxton [15] concretized the concept of three state model of input in which states are sensed by the input device. These states are tracking; which allows cursor to move, dragging; which allows the user to click and drag an object and out of range, which happens when the input device moves out of the physical range (eg., lifting stylus from tablet, lifting mouse from mouse pad). Usually pointing devices like mouse sense only two states mainly tracking and dragging. Devices like touch pad senses tracking and out of range states. Hence to map functionality of mouse to a touch pad one has to add features like single tap for clicking. This also creates a problem of accidental click trigger in case of the touch pad.

2.1.4 Direct and Indirect Control

The direct input control occurs when the input space coincides with the output space (eg., stylus and tablet, touch screen). In contrast, the input space does not coincide with the display (eg., mouse, joystick) in case of indirect input control. Hinckley, et al. [1] outlines the drawbacks of direct input control which includes parallax error, reduced transmissivity due to sensing layer, finger occlusion during input. Apart from these the touch screen can support cursor tracking or cursor dragging state but not both.
2.1.5 Device Acquisition Time

Hinckley et al. [1] defines *device acquisition time* as the average time to pick up or put down an input device. Though stylus or pen-based input devices can perform at the same level as a mouse device for pointing tasks [16] or better than a mouse for some high precision tasks [17], it takes more time to switch between pen and keyboard devices which are not desirable. This can be more understood in terms of CAD application interface where users have to perform 3D modeling operations and access keyboard after each operation to input the dimensions.

2.2 Pointing Devices

2.2.1 Mouse

The mouse senses the relative motion on a flat surface. It has two degree of freedom mainly horizontal and vertical. It is an indirect input device. Hinckley, et al. [1] placed emphasis on the stability of mouse (mouse does not fall as stylus or pen) and that the force required to activate the mouse button is orthogonal to the plane of movement which minimizes accidental clicking or interference with the motion. Moreover mouse are well suited for desktop GUI [18] and Fitts law studies showed that users can point with the mouse as good as with hand [19].

2.2.2 Trackball

A trackball has a mechanical ball to roll the position of pointer while the device itself has no relative motion with the surface on which it is positioned. The main advantage of the trackball is that it can be used on an inclined surface. The major disadvantage of the trackball is that it is uncomfortable for press and drag tasks where the user has to press the button and roll the ball itself [20].
2.2.3 Tablets

Tablets are used for tasks such as tracing, freehand drawing and signature tracking. It is a device that relies on absolute position of input tool such as stylus or a puck. The common problem with tablet is that it is responsive to stylus alone and does not allow any other tool. Touch tablets are the devices that register finger touches. The common application of touch pads can be seen in portable computers (laptops). A touch screen is a transparent layer that is mounted on display device to sense the touch inputs.

2.2.4 Pen Devices

Pen based input device are used for marking, inking and gestural inputs. Hinckley, et al. [1] addresses the problem of pen input device when used in conjunction with normal GUI that is designed for mouse. One of the major problems is that the user cannot see an entity before selecting it. In other words the pen input device directly supports dragging state, eliminating hover or tracking state. Apart from these drawbacks, the pen devices, like a stylus use inductive sensing technology which inhibits the user from using the pen on any other surface. As Hinckley pointed out, the inductive sensing enables the screen to track the pen when it is close to it and not touching it, thereby giving an advantage of incorporating tracking state as well.

2.2.5 Joystick

As mentioned earlier, the joystick measures force and returns to its original position once the force is removed. An isometric joystick (like the ones present on an IBM laptop) is small in size and hence can be fit anywhere near the keyboard. An IBM laptop has an isometric joystick which is placed approximately at the center of the keyboard which gives and excellent chance to switch between pointing and typing tasks [21] [22]. In other words it has small device acquisition time. Isotonic joystick senses the angle of deflection of the stick and tends to move more than
isometric joystick providing better feedback to the users [1]. They can be designed with mechanical springs to bring them back to the center automatically. Some joystick may have both force and position sensing abilities [1].

2.2.6 Keyboard
A keyboard is mainly used for typing and text editing. Apart from these tasks they are used to aid easy and quick task completion by incorporating shortcut key combinations, navigation keys, etc. A procedural memory stores repetitive motor actions [1]. After storing, it requires the user to use a less conscious effort to perform a repetitive task [23]. Power law of practice \( T = aP^b \), \( T \) is the time to perform task, \( P \) is amount of practice, \( a \) and \( b \) are constants) reflects the procedural memory encoding. Because of this reason the keyboard layout cannot be changed without having significant re-learning cost. Lewis, et al. [24] brought to attention that with practice users can memorize the position of keys and can type without any visual attention. This of course is dependent on home position of the two hands. In recent years, devices like apple i-phone, i-pad and many other mobile devices have implemented soft keyboards into their system. Zhai, et al. [25] outlines that the design issues with that of soft keyboard are very different from that of mechanical keyboard. Mechanical keyboard have tactile feedback, i.e., the users can feel the physical keyboard and the keys provide a feedback each time they are pushed. A soft keyboard lacks such a feedback system and hence its performance is effected [24].

2.2.7 Gesture Based Input
Gesture is a sign or a symbol performed by the user to communicate a specific command to the computer. Pen based devices can be used for sketch editing tasks such as erase (rapid zig-zag gesture like erasing action) or circling an entity and dragging it in order to copy that entity; but the difficulty lies in the inability of the computer system to understand the user actions. The
system confusion lies in treating the user action as a command or not. Consider an example of a gesture-based pen input system in which the user has to draw zig-zag lines to erase an entity. If the user wants to draw a zig-zag line for sketching and not for erasing then the system gets confused between the two actions. In spite of this drawback, people have tried to use the pen input to define 3D objects by means of sketching [26] [27]. Finger-based input is a more natural way of interaction than pen-based input. Hand gestures can be broadly classified as semiotic, ergotic and epistemic [28]. Semiotic gestures convey useful information like nodding one’s head for indicating “yes”. Ergotic gestures are used for manipulating physical objects. Epistemic gestures are movements to explore and acquire haptic and tactile feedback. The most research is done in semiotic gesture area. One can refer to Rime, et al. [29] for further classification of semiotic gestures.

2.2.8 Multitouch Devices

Multitouch devices can be classified as a direct input type since both output and input areas are same. The finger touch (single finger, multiple fingers) is the primary means of input. These devices also support multiple users. The advent of multitouch interface has diminished the gap between input and output devices. This novel interface was made popular by Jeff Han [30] who introduced FTIR multitouch table at the TED talks. The amazing aspect of this launch was not the hardware but the new found applications that can be subjected to finger gesture manipulations. For a detailed description of multitouch devices, see Brandl, et al. [32] and Muller [33].
Chapter 3

Hardware Design of Multitouch Interface

The previous chapter shed some light on various types of input devices. In this chapter we will discuss more about a novel interface called as “Multitouch interface” that has recently gained some popularity. Multitouch, as the name suggests is a device that enables a single user or multiusers to input using multiple fingers or even objects. Research in multitouch interaction has been around since 1980. Bill Buxton [31] considered by many a pioneer in this area. He has contributed to the area of multitouch research more than 20 years. This interface gained popularity since the introduction of FTIR based multitouch system by Jeff Han in 2006 [30]. The very next year apple launched the multitouch-enabled mobile phone device called i-phone which has become ubiquitous in the world of multitouch-enabled smart phone market. Since then many companies have introduced their version of multitouch-enabled devices. Commercially available multitouch devices can be categorized as:

- Personal computers and laptops (Dell, HP Touchsmart)
- Multitouch table (Microsoft Surface, MERL DiamondTouch)
- Multitouch-enabled smart phones (i-phone, Google TI)
- Multitouch-enabled utility devices for performing specific tasks (Wacom Bamboo)

3.1 Classification of Multitouch Hardware

The multitouch hardware system alone is divided into many categories. Brandl, et al. [32] gave a good introduction on existing multitouch technologies and categorizes them as follows:
• Resistance based touch surfaces
• Capacitance based touch surfaces
• Surface wave touch surfaces
• Optical based touch surfaces

One can refer to Brandl, et al. [32] for more information on resistance, capacitance and surface wave touch surfaces. Optical based touch surfaces are popular and cost effective and hence we will discuss them in detail.

3.1.1 Optical Based Touch Surfaces

Due to their simplicity, cost effective construction, scalability and robustness, optical-based multitouch surfaces are the favorite choice across industries and research group. Both optical and camera-based approaches share the same concept of processing and filtering captured images on patterns. The Figure 1 displays the camera based multitouch system pipeline. A human subject finger touches are recorded by a camera. The camera not only captures the touch points but also the static background. Hence there is a need for image processing to extract the touch points from the background. The camera captures the current frame which is converted into greyscale image and the background is removed by subtracting the current frame from a reference frame. This results in the isolation of the touch points that takes the form of white contours which are termed blobs. Touch sensing is thus achieved by tracking the blobs. A blob tracker tracks the position of the blobs by comparing the current frame of blobs with previous frame of blobs. Thus the blob positions and the change in their positions can be used to convey some meaningful information by treating them as gestures. The application responds according to the programmed gestures. The results are beamed back using a projector to the input surface. Hence this qualifies
as a direct type of input space where the input and the output area are the same as discussed in the section 2.1.4.

The performance of a camera-based multitouch system depends on both hardware and the software. Since the pipeline contains many elements, the responsiveness of the system depends on all the process is from detecting touch points to display the results. Usually the commercially available camera can support 30 frames per second to ensure smooth interaction; but fast software and hardware processing capabilities are required to increase the responsiveness and minimize the latency.

![Optical based multitouch tracking pipeline](33)

**Figure 1:** Optical based multitouch tracking pipeline [33]

We now take a look at existing optical-based multitouch systems and discuss their advantages and disadvantages.
Frustrated Total Internal Reflection (FTIR)

The rediscovery of the Frustrated Total Internal Reflection (FTIR) principle by Han [30] in 2005 can be seen as a starting point for optical multi-touch systems. FTIR technology is based on optical total internal reflection within an interactive surface. This device works on the concept of Total Internal Reflection (TIR). Total internal reflection is an optical phenomenon that occurs when a ray of light strikes a medium boundary at an angle greater than a particular critical angle with respect to the normal to the surface. If the refractive index is lower on the other side of the boundary, no light can pass through and all of the light is reflected. The critical angle is the angle of incidence above which the total internal reflection occurs. In case of FTIR system, Infrared rays are introduced into an acrylic sheet from its edges. The acrylic traps the infrared rays inside the acrylic (see Figure 2). So when a user touches the acrylic plane, the change in refractive index will cause the infrared rays to bounce inwards towards the camera. The camera picks up the touch blobs. For the camera to detect infrared blobs it must have an infrared band-pass filter, which will allow only infrared rays to pass through. From Figure 2 one can also see a diffuser material. The diffuser material has two main functions (1) to block the camera from seeing the unnecessary background image, (2) to act as a projection screen for the DLP projector. Muller [33] claims that the performance of the system depends upon how greasy the touch surface is, i.e., wet fingers have better contact than dry fingers. Hence for better contact detection, an alternative FTIR layout is described by Muller [33]. Figure 3 shows the alternative FTIR layout. In this setup a compliant layer is sandwiched between acrylic and the diffuser sheet. Van der Veen [34] suggested that the compliant layer can be made of silicon material (like ELASTOSIL M 4641). On top of the compliant layer there is diffuser material (like Rosco Grey). The diffuser layer makes it possible to send a clean image to the camera so that only few image processing
19 steps are required. The function of the baffle, (see Figure 2 and 3), is to prevent leakage of the infrared rays from the edges.

Figure 2: FTIR setup without Compliant layer [33] and the corresponding finger blob contrast.

Figure 3: Alternate FTIR setup with Compliant layer [33].
**Diffuse Illumination (DI)**

**Rear Side Illumination**

This system is used in products such as Microsoft Surface and HoloWall. A diffuser layer is put beneath a glass sheet or an acrylic sheet, (see Figure 4). Position right below the glass sheet, infrared illuminators projects the infrared rays on to the diffuser. Some of the infrared rays will be diffused while others will pass through the layer. When the fingertip touches the display, the infrared rays are reflected back to the camera; thus the touch points are detected. There are a few advantages and disadvantages for this system as compared to FTIR setup. Since RI system banks on reflection rather than on TIR principle, it works well with both wet and dry contact. Moreover it can track fiducial markers [35] which enable the system to recognize objects; its position and orientation. However the major problem for RI system lies in the diffuser material. The diffuser material should not absorb much infrared rays. In such a scenario, the contrast of the touch blobs is reduced; it makes harder for the system to recognize them. If the diffuser absorbs only a few infrared rays, then it allows more infrared rays to pass through; this causes detection of a finger when it is not touching but it is just near the surface. This is termed as false blob. Apart from this there is also the problem of uniform illumination. One has to use multiple infrared illuminators for RI system and in doing so the uniformity of the diffusion of the infrared rays is disturbed. Some parts of the surface may have bright infrared spots while other parts suffer from low illumination. Since RI system relies heavily on contrast difference, it cannot be used in direct sunlight. Moreover the RI system requires more image processing steps than FTIR system.
Figure 4: Rear side illumination setup [33] with corresponding finger blob contrast

Front Side Illumination

A front side illumination system relies on ambient light than on infrared light source. In this system the diffuser is placed on the top of the acrylic which diffuses the ambient light. When a user touches the screen, a shadow is created beneath the finger touch area (see Figure 5). This shadow is detected by the camera as a touch point. Since it is necessary for the touch point to be white in color, a simple invert filter is applied to the shadow image to make it white in color. This system again suffers from uniform illumination problem and thereby is less reliable and less precise than a FTIR or RI system.

Figure 5: Front side illumination setup [33]
Diffused Surface Illumination (DSI)

Diffused Surface Illumination (DSI) addresses the problem of getting an even distribution of infrared light across the screen surface. This problem is tackled in DI set-ups by using a small number of (two or three) Infrared Illuminators. Tim Roth [36] proposed the use of a special acrylic that incorporates small particles that act as tiny mirrors. So when IR light shines into the edges of this material it is redirected and evenly spread across the surface. Figure 6 illustrates the DSI set-up.

![Diffused Surface Illumination Setup]

Figure 6: Diffused surface illumination setup [32] with corresponding finger blob contrast

Selecting the setup

A quick comparison as suggested by Muller [33], will allow users to select the type of optical based multitouch system suitable for their application. Table 1 shows the comparison of different optical-based multitouch system.
Table 1: Comparison of different optical based multitouch system [33]

<table>
<thead>
<tr>
<th>Factors</th>
<th>FTIR</th>
<th>RI</th>
<th>FI</th>
<th>DSI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>High</td>
<td>Medium</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td>Construction complexity</td>
<td>High</td>
<td>Medium</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td>Closed box required</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Blob Contrast</td>
<td>Strong</td>
<td>Average</td>
<td>Average</td>
<td>Strong</td>
</tr>
<tr>
<td>Software tracking complexity</td>
<td>Low</td>
<td>Medium</td>
<td>Medium</td>
<td>Low</td>
</tr>
<tr>
<td>Reliable finger tracking</td>
<td>High</td>
<td>High</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Allows object tracking (Stylus)</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Allows fiducial tracking</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Influence of ambient light</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>Low</td>
</tr>
</tbody>
</table>

A broader comparison of different optical-based multitouch systems is given in Table 2.

Table 2: Broad Comparison of the different optical based multitouch system [37]

<table>
<thead>
<tr>
<th>Touch Technology</th>
<th>DI</th>
<th>DSI</th>
<th>FTIR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technology</td>
<td>Image analysis</td>
<td>Image analysis</td>
<td>Image analysis</td>
</tr>
<tr>
<td>Activation</td>
<td>Zero activation force required</td>
<td>Zero activation force required</td>
<td>Zero activation force required</td>
</tr>
<tr>
<td>Transmissivity/optics</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Drag and Drop</td>
<td>High resolution, good drag and draws smooth lines</td>
<td>High resolution, good drag and draws smooth lines</td>
<td>High resolution, good drag and draws smooth lines</td>
</tr>
<tr>
<td>Calibration</td>
<td>Requires periodic re-calibration</td>
<td>Requires periodic re-calibration</td>
<td>Requires periodic re-calibration</td>
</tr>
<tr>
<td>Surface Contaminants/Durability</td>
<td>Potential for false activation and dead zones from surface contaminants</td>
<td>Potential for false activation and dead zones from surface contaminants</td>
<td>Potential for false activation and dead zones from surface contaminants</td>
</tr>
<tr>
<td>Sensor Substrate</td>
<td>Acrylic or glass</td>
<td>Acrylic or glass</td>
<td>Acrylic</td>
</tr>
<tr>
<td>Single/dual/multitouch</td>
<td>Multitouch</td>
<td>Multitouch</td>
<td>Multitouch</td>
</tr>
<tr>
<td>Touch Points</td>
<td>Unlimited, but computer performance dependent</td>
<td>Unlimited, but computer performance dependent</td>
<td>Unlimited, but computer performance dependent</td>
</tr>
<tr>
<td>Display Size</td>
<td>Dependent to technique used</td>
<td>Dependent to technique used</td>
<td>Dependent to technique used</td>
</tr>
<tr>
<td></td>
<td>(projector of LCD)</td>
<td>(projector of LCD)</td>
<td>(projector of LCD)</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------------</td>
<td>--------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>Size Constraints</td>
<td>Scales to larger sizes</td>
<td>Scales to larger sizes</td>
<td>Scales to larger sizes</td>
</tr>
<tr>
<td>Sealable</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Integration</td>
<td>Total setup only</td>
<td>Total setup only</td>
<td>Total setup only</td>
</tr>
<tr>
<td>Touch Method</td>
<td>Can use any pointing device</td>
<td>Can use any pointing device</td>
<td>Can use any pointing device</td>
</tr>
<tr>
<td>Touch Accuracy</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>Drivers</td>
<td>HID compliant no additional drivers required</td>
<td>HID compliant no additional drivers required</td>
<td>HID compliant no additional drivers required</td>
</tr>
<tr>
<td>Limitations</td>
<td>Special surface coating required, sensitive to ambient lighting conditions</td>
<td>Special surface coating required, sensitive to ambient lighting conditions</td>
<td>Special surface coating required, sensitive to ambient lighting conditions</td>
</tr>
</tbody>
</table>

3.2 Choosing the Components for Optical Based Multitouch System

3.2.1 Camera

Any camera-based sensor requires a camera capable of detecting infrared light (except Front DI). The camera has an image sensor (CMOS/CCD) which has varying sensitivity to infrared rays. Usually the infrared spots are detected as a bright white spot or with blue/purple glow [33]. When choosing the camera it is important to refer to the spectral characteristics of the camera to find out how sensitive it is to infrared light.

A commercially available consumer camera is most often used for a FTIR or RI system; but these cameras have an infrared blocking filter that prevents them from detecting the infrared rays. The simple solution is to remove the filter and replace it by infrared bandpass filter. IR bandpass filter allows only infrared rays to pass through. It is important that the bandpass selected must have spectral characteristic to allow IR rays from the IR illuminators used in the system. For example the most common IR illuminators have wavelength of 850nm to 880nm.
The bandpass filter must be sensitive to the wavelength of the illuminators. 780nm, 880nm, 940nm are the other wavelength ranges of the IR illuminators that can be used for same purpose.

Another low cost alternative to IR bandpass filter is a floppy disk film or a developed film negative that has been exposed to sunlight. The performance of these low cost alternatives is affected by ambient light which makes it very hard to place the multitouch systems outside a dark room. One layer of exposed film negative acts as a longpass than a bandpass. Two layers of exposed film negative have better filtering capability, but they have low transmission properties. One layer of floppy disk filter has low transmission and bad filtering capabilities. Moreover it does nothing to prevent visible light. Finally two layers of floppy disk layer are almost opaque. These low cost filtering options usually do not have necessary filtering properties and hence it is important that one chooses an IR bandpass filter.

Frames per second (FPS) is an important parameter for selection of the camera; 30 fps is recommended for smooth interaction [33]. A camera like Sony PS3 can support 60 to 120 fps, but a high fps camera requires the software that can process information at high speeds.

### 3.2.2 DLP Projector

Projector plays an important role in the performance of the optical/camera-based multitouch system; 1024 X 768 pixels resolution is recommended irrespective of the applications [33]. Digital Light Processing (DLP) and Liquid Crystal Display (LCD) can be selected based on contrast ratio and lumen brightness. Since these two parameters are important in case of RI system which relies heavily on contrast and reduced brightness. One of the major factors that come into picture is the throw of the projector. The two types of throw include 1) Short throw and 2) Long throw.
Throw ratio [40] is defined as $T = D/W$. By considering the value of $W$ to be constant, the short throw ratio yields a smaller value than long throw ratio. Projectors used in offices are usually long throw. The long throw projectors cannot be directly used for multitouch systems since they need a large $D$ to project correctly. The major modification for using long throw projector is to introduce a mirror and reflect the projector light on to the display screen. Mirrors help to cut down the projection distance, but at the cost of quality and brightness. Using long throw projectors in combination with mirrors is a cost effective solution since short throw projectors are very expensive.

### 3.3 Optical Based Multitouch Image Processing Software

There are open source software like touchlib [38] and CCV [39] that provide image processing and blob tracking facilities. Both CCV and touchlib have similar image processing, blob detection and blob tracking systems. In case of FTIR-based setup the image processing steps,
depending upon the noise, are short. The first frame is captured at the start of CCV application. This is considered as a reference frame and stored in the memory. From this reference frame all the background subtraction processes are initiated. The background filter eliminates the background and the rectify filter reduces the noise and converts the grey scale image into a black and white image (see Figure 8).

Figure 8: CCV images of a FTIR setup

In case of an RI system the image processing involves additional steps (see Figure 9). The process starts when the capture filter retains the current frame as the reference frame. Then the background filter is applied to remove any static background elements. The high-pass filter then selects only the high contrast region to pass through. The values of high-pass can be defined by the user. The contrast of the blobs depends upon the diffuser and the strength of the IR source. Hence the amplify filter intensify the brightness of the blobs. Finally the rectify filter converts the grey scale image into a black and white image with reduced noise.
According to Muller [33], the camera has intrinsic and extrinsic properties. Intrinsic properties include focal length, image center, etc. Extrinsic properties reveal the relationship between camera and the world like rotation matrix, translation vector, etc. Before application of the rectify filter another filter called as barrel distortion correction filter is applied for removing the lens distortions. For the mathematics of removing distortion one can refer to Muller [33].

### 3.3.1 Blob Detection

The first step prior to blob tracking is blob detection [33]. The CCV requests frames from the video and for each frame it stores the contour found in it. Each contour is then checked if it is a fiducial marker [35] or a touch. After this step each contour is fitted with a polygon in order to match the edges of the contour. If the resulting polygon fit has four points then the system may consider it as a fiducial marker. The system then confirms this fact if the resulting corners yield
90 degree angle. Then the center of fiducial marker is calculated and its orientation is registered. In case the polygon has more than four points, the system tries to fit an ellipse around the contour and calculates the center. The properties of the ellipse are used to determine the position and orientation of the blob. The blob is registered and given an ID if it lies between the minimum and the maximum size limit. These limits are defined by the user.

![CCV images of Blob Detection](image)

**Figure 10:** CCV images of Blob Detection

### 3.3.2 Blob Tracking

The blob tracking system is well explained by Muller [33]. The details on blob tracking process are as follows:

The system stores the set of $k$ active blobs detected from the previous frame. This forms the first data set.

$$A_1, A_2, A_3, \ldots, A_k$$

The second data set stores $l$ active blobs detected from the current frame, (see Figure 11).

$$B_1, B_2, B_3, \ldots, B_l$$
For each passing frame, the data set $A$ is replaced by data set $B$ and new set of blob data are filled in data set $B$.

![Figure 11: Blobs detected in the previous frame (left) and current frame (right)](image)

Previous frame data set:

$A_1(2,3), A_2(1,5) \quad k=2$

Current frame data set:

$B_1(1,3), B_2(4,2), B_3(5,3) \quad l=3$

There is a need for creating a matrix that retains all possible states in order to match blobs from the previous frame and the current frame. The number of possible states is calculated using the formula given below:

$$ N = \frac{(k-x)!}{x!(k+x-l)!} $$

Where $x = l-k$

Hence for the above example $N$ yields a value of 6. For each possible state an entry is added to the transition matrix (see Table 3).
Table 3: Transition matrix indicating all possible states

<table>
<thead>
<tr>
<th>States</th>
<th>$B_1$</th>
<th>$B_2$</th>
<th>$B_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>New</td>
<td>$A_1$</td>
<td>$A_2$</td>
</tr>
<tr>
<td>$S_2$</td>
<td>New</td>
<td>$A_2$</td>
<td>$A_1$</td>
</tr>
<tr>
<td>$S_3$</td>
<td>$A_1$</td>
<td>$A_2$</td>
<td>new</td>
</tr>
<tr>
<td>$S_4$</td>
<td>$A_1$</td>
<td>new</td>
<td>$A_2$</td>
</tr>
<tr>
<td>$S_5$</td>
<td>$A_2$</td>
<td>$A_1$</td>
<td>New</td>
</tr>
<tr>
<td>$S_6$</td>
<td>$A_2$</td>
<td>new</td>
<td>$A_1$</td>
</tr>
</tbody>
</table>

After the transition matrix is stored, the system tries to find a state which yields the lowest distance value. The system assigns a zero value for the new blob. As seen from the above example, the current frame contains more data than the previous frame, which means that a new blob is present in the current frame. The distance values are calculated from the transition matrix as follows:

$$S_1 = 0 + \sqrt{(1-4)^2 + (5-2)^2} + \sqrt{(1-5)^2 + (5-3)^2} = 4.24 + 4.74 = 8.98$$

Similarly all the distance values for all the states are calculated. The system then chooses the row with lowest distance value. In case of two lowest distances having the same value, the system chooses the first row with that value. The final tracking of the system is shown in Figure 12.
3.3.3 Generating Events

After detecting and tracking blobs, the system needs to generate events based on the finger touch. The system scans the current frame blob list and compares it with the previous frame blob list. In the event of the existence of a new blob in the current frame, the system assigns the blob ID a value of -1 [33]. For every new ID generated a touch event is dispatched. If the blob in the current frame matches the blob in the previous frame then the same blob ID is retained. An update event is dispatched depending upon the blob movement. If the delta movement of the current frame blob and with respect to the previous frame blob is higher than the set threshold distance then instead of update event a touch event is generated. If the delta movement is less than the minimum threshold distance then no update event is dispatched. Whenever a blob in the

Figure 12: Final results of the blob tracking process
current frame is missing as compared to that of the previous frame, the blob is marked for deletion and a touch up event is dispatched.

### 3.3.4 Communicating the events

Once the events are generated it is important for the system to transfer the blob data so that it can be accessed by programming languages and used by applications. Touchlib and CCV contains a wrapper which sends TUIO events over Open Sound Control (OSC) system. The OSC system uses UDP network protocol to transfer the blob data. Programming languages like Python, Java, C#, Action script have OSC libraries that support the data transfer. Flash interface cannot process UDP package and hence need a proxy step to convert UDP into TCP. This proxy step is called as Flosc, see figure 13. The important point to remember is that whenever an application uses UDP protocol, it can only receive data pertaining to blob properties and hence cannot make any changes in touchlib or CCV settings.

![Schematic view of the OSC communication pipeline](image)

**Figure 13:** Schematic view of the OSC communication pipeline [33]
Chapter 4

Evaluation Techniques of Input Devices

4.1 Fitts Law

In 1954 Paul Fitts [41] of Ohio State University created a model of human movement which predicts the movement time between a start and target object with a specific distance. Fitts implemented this experiment by creating a board with adjustable metal plates. The test subjects were asked to perform a one dimensional tapping test between the two metal plates by using a stylus (see Figure 14). The experiments were performed on the metal plates of different widths (W) and different inter-plate distances (A). During the test the apparatus recorded the time difference between touching of the two metal plates. In order to measure the accuracy, Fitts added two extra plates on both sides of the metal plates in order to record a miss.

![Fitts reciprocal tapping device](image)

**Figure 14:** Fitts reciprocal tapping device [41]
Mathematically Fitt’s law is represented as:

\[ MT = a + b \log_2 \left( \frac{2A}{W} \right) \]  

\[ ID = \log_2 \left( \frac{2A}{W} \right) \]

Where \( MT \) is the average time to perform the task (movement time); \( a \) and \( b \) are constants which depend on the type of user and the type of input device; \( A \) is the distance between the start object and the target object measured from the center of both objects; \( W \) is the width of the object in horizontal axis; and \( ID \) is the index of difficulty.

The constant \( a \) describes the cost of time that is spent to perform nonphysical movement such as the processing time for recognizing objects. The constant \( b \), which is a time unit and often measured in milliseconds, is multiplied with \( ID \). In order to compare the pointing performance of an input device, it is required to characterize the performance into one value. This value is called the index of performance (\( IP \)). Two common methods to calculate this value are given in Equations (2) and (3)

\[ IP = 1/b \]  

\[ IP = \frac{ID}{MT} \]

In Equation (3), the impact of constant \( a \) is ignored. The performed measurements can be used in the first equation. There are instances where the Equation (1) gives a negative value for a
difficult task. This occurs when the amplitude is less than half the target width. To solve this issue MacKenzie [42] proposed the equation known as the Shannon formulation:

\[
MT = a + b \log_2 ((A/W) + 1)
\]

(4)

\[
ID = \log_2 (\frac{A}{W} + 1)
\]

(5)

By using the Shannon formulation we preserve the characteristics of Fitts law and at the same time presented a result with a positive rating always. Hence this formulation is commonly used.

The importance of Index of performance is stated by Bill Buxton [43]. The major advantage of \textit{IP} is that (1) It provides a metric for quantifying and comparing the capacity of fingers, arm and wrist in the performance of various tasks, (2) it provides a way for quantifying actual performance so that various input devices and their respective techniques can be compared.

4.2 Fitts Law for Two Dimensional tasks

The experiment conducted by Fitts involved measuring performance between two horizontally separated targets. This qualifies as a one-dimensional task; but the direct manipulation of the objects involves performing task in two-dimensional or three-dimensional domain. The major problem of applying Fitts law to 2D tasks is the selection of target width and amplitude (see Figure 15). Card [19] and Jagacinski and Monk [44] have applied Fitts paradigm for 2D pointing tasks. Buxton [43] states that the researchers face ambiguity by applying 1D Fitts model for 2D movement tasks where the target width is the horizontal target and the target amplitude is the distance to the center of the target (see figure 15). In the case of a 2D operation two major factors that must be considered are (1) effect of angle of approach on the performance, (2) what is the width of when the target is asymmetric and approached form different angles? [43]
Studies performed by Card [19] show that for 2D pointing tasks, the approach angle has very little impact. In case of the mouse there was no difference in the diagonal movement time as compared to vertical and horizontal movement time. In case of joystick there was 3% increase in diagonal movement time. The second problem is the calculation of the target width. As can be seen from Figure 15, for an asymmetrical target one faces the question what should be considered as width? \( W \) or \( W' \). Gilan, et al. [45] investigated this problem by considering various alternatives such as \( H \): Target height, \( W \): Target width, \( H + W \): target height + width, \( H \times W \): Target area – (height times width). Buxton [43] points out that their method of defining underlying tasks did not conform to Fitts paradigm and hence the results were problematic. Mackinzie [42] examined the following parameters to calculate the effective target width and they are listed as follows:

- \( W \): target width
- \( W + H \): target width plus height
- \( W \times H \): target area – (height times width)
- \( SOWH \): the shorter of width or height
- \( W' \): projecting the approach vector through the target, the length of the portion intersecting the

**Figure 15:** Determining effective target width [42][43]
Form the studies both $W+H$ and $W \times H$ are discarded for being no better than $W$. $SOWH$ and $W'$ were observed to be superior to $W$ model; they did not differ much from each other. $SOWH$ is computationally more efficient than $W'$ but $W'$ works much better for non-rectangular targets and preserves the one dimensionality of the Fitts model.

Usually Fitts law is used for pointing tasks. But it can also be extended to dragging tasks. Gillan, et al. [45], Mackenzie [42], Sellen and Buxton [20] have used Fitts law for evaluating dragging tasks. For experiments on dragging tasks, refer to Buxton [43].

### 4.3 Steering Law

Fitts law is generally used for pointing and dragging tasks. However there are other types of interactions like steering in HCI which has to be considered. Steering is moving along trajectories on a computer screen [43]. The best example of steering is moving through nested menus to select a particular entity. Drawing curves can also be classified as steering. Due to the different nature of steering it cannot be expected to follow Fitts law. Accot and Zhai [46] discovered a law linking steering difficulty and steering speed and hence connecting steering law with Fitts law. Their theory confirmed that passing through Goals follows Fitts law (see Figure 16). In this case the width $W$ is perpendicular to the direction of the movement of the cursor. Fitts law needs the cursor to stop when it reaches the target while steering requires users to pass through the goals. Accot and Zhai [46] confirmed that these tasks follow the same speed vs. accuracy relationship as Fitts law.

$$MT = a + b \log_2 (A/W+1)$$

Where $MT$ is the time taken to pass through the goals; $A$ is the distance between the goals; and $W$ is the width of each goal.
Accot and Zhai [46] placed $N$ number of goals along the trajectory and established that the index of difficulty changes from $\log_2 (A/W+1)$ to $N \log_2 (A/NW+1)$.

If $N$ is varied to infinity then goal passing task will become a steering task. This will mean that the index of difficulty changes to $A / W \ln (2)$. Combining the $\ln (2)$ with constant $b$ we get

$$MT = a + b \frac{A}{W}$$

The steering law was then extended to different shapes of tunnel mainly conical and spiral (see Figure 17). The generalized formula for steering law is given as:
\[ MT = a + b \int_c^e \frac{1}{W(s)} ds \]

Where \( a \) and \( b \) are constants that depends upon the shape of the tunnel, the type of the device used and the user; it is similar to Fitts law. For a same steering tunnel shape and a same group of users \( b \) reflects the control quality of the input device [43].

**Figure 18:** General steering law
Chapter 5

**Key Elements of the Multitouch Interface**

From Chapters 2 through 4, we have discussed about input devices, types of multitouch hardware and evaluation and comparison of the multitouch device and the mouse device. This chapter has the following objectives: (1) identifying the key elements for designing a good multitouch software interface, (2) identify the key multitouch hardware issues during interaction, (3) outline the computer aided design (CAD) process and form a blue print for amalgamation of CAD and the multitouch interface.

5.1 Multitouch Interface Research

![Diagram of Key Elements of a Multitouch Interface]

**Figure 19:** Schematic diagram showing the key elements of a multitouch interface
Figure 19 divides the research area of multitouch interaction into hardware and software categories. We will initiate the discussion focusing on multitouch software application interface.

5.2 Multitouch Software

5.2.1 Touch Input Properties

The multitouch relies on finger touch input for the primary source of input. The conventional mouse has three buttons which can be used in various combinations for triggering events. When it comes to finger touch, we have an option of ten fingers and their combinations. But the combinations decrease since one may not be comfortable with using all ten fingers. Moreover the factor of dominant hand plays a major role in diminishing the available finger combinations. This requires developers to tap into various finger touch properties that can be used as alternative inputs. The various finger touch properties are described in the section below.

**Finger Touch Input**

Studies conducted by Dandekar, et al. [47] suggested that adult fingers generally have a diameter of 16mm to 20mm (0.6" to 0.8"). When interacting with a touchscreen, users will prefer to use the pad of their finger rather than the very tip of the finger. The pad of the finger is slightly narrower than the full width of the finger: 10-14mm. The fingertip is smaller: 8-10mm wide. In current multitouch systems the finger touch is mainly used for positioning cursor, click and drag events. When a finger touches the screen, the X and Y coordinates of the contact area is given as the output. This limits the degree of freedom from 23 to 10 (considering 5 fingers). Wang, et al [48] suggested that various physical finger input properties as show in Figure 20 can be exploited by designers for different inputs.
Current multitouch applications consider distance between two points for triggering an event. The best example of this technique is the Zoom gesture shown in Figure 20. Hence considering other physical finger input parameters is essential to increase the options of gestures available for developers. As per Wang, et al. [48], the other finger properties that can be exploited are given as follows.

**Position property**

This type of input property considers the finger property as co-ordinate values (X, Y). The most common property used is finger contact. This property was first studied by Buxton [49][50] and Lee[51]. The center point of each contact area is considered as a touch point. This enables the users to interact with tradition GUI elements and perform actions such as click, drag and drop.

**Motion property**

The Motion property considers the velocity and the acceleration of the finger movement. The movement vector and motion acceleration was first used by TUIO protocol [52]. This property is also widely used as a gesture for finger interaction. Some of the famous applications include this property to throw pictures in the direction of motion of the finger across large multitouch displays.
**Physical property**

The physical properties of finger touch are mainly categorized as (1) size of contact area, (2) shape of contact area, (3) orientation, and (4) pressure. These properties are less used. Contact area and pressure based properties were studied by Benko, et.al. [53]; they used finger rocking and pressing to trigger click events. A Pressure widget uses the pressure sensing abilities of stylus interface to operate widgets. Forlines, et al. [54] studied the shape variation properties of the finger touch; they studied the vertical and oblique finger touch techniques and their relation to selection errors. The orientation property has been recently studied by Wang, et al. [48] The major limitation of the use of change of contact area property and orientation property as reported by Wang, et al. [48] is that multiple fingers cannot be used to trigger events since for all fingers area of contact and orientation changes from finger to finger.

**Event property**

This property is used for triggering events based on finger gesture. For example a single finger tap is used as a mouse “click”. The popular i-phone zoom gesture that scales pictures according to the change in distance between two touch points.

**5.2.2 GUI Element Design**

Apart from finger properties, the GUI interface design should be given much focus. Though the finger touch forms the main bulk of input, users generally interact with a system via conventional GUI elements such as buttons, icons, menus, etc. Hence the GUI interface must be designed for precision and occlusion avoidance.
**Touch Precision**

Precision is a major issue when it comes to finger touch. A mouse pointer is of size 16 X 16 pixels and hence is convenient to select smaller targets with high precision. In contrast, the finger pad dimensions are much bigger as discussed in section 5.2.1. Another problem is that a finger does not float transparently in space like a mouse pointer does. With multi-touch screens, the user will not be able to see the part of the screen covered by the finger; this makes the selection of an entity difficult. Hall, et al. [55] found that accuracy varied from 66.7% for targets of 10 mm per side to 99.2% for targets of 26mm per side. This study was limited only to the index finger. Wang, et al. [48] examined the finger touch precision of all fingers and recommended that in order to maintain direct touch precision, the touch targets must be at least 14.38 pixels (5.76mm) in radius for circular targets and 28.76 pixels (11.52mm) per side for square targets. The experimental results of Wang were similar to that of Shneiderman [56]. They reported that the maximum accuracy is achievable when the targets are 32 pixels per side (13.8 x 17.9 mm). The UME guidelines [57] also suggest that the interface elements should not be less than 10 mm each side. These guidelines help users to increase the selection precision.

**5.2.3 GUI Elements Layout**

Designing the GUI elements for precision alone is not sufficient. There is a major problem of finger and hand occlusion when it comes to multitouch interaction. Albinsson, et al. [58] reported that the arm, finger, hand occlusion included significant screen area which eventually led to low selection precision. This study established a relation between occlusion and selection precision. Hence the developers, concentrating for selection precision must focus to reduce the occlusion problem. The vast majority of the engineering application software found today is
designed for mouse and hence follows similar rules. They place icon toolbar and menu at the top area of the window. Considering the perspective distortion problem (will be discussed in the section 6.2) and the arm occlusion; this forms a bad design for finger touch interaction. Moreover applications CAD and other engineering applications rely heavily on a visual feedback system while performing specific tasks. This visual feedback does not reach the user due to hand occlusion. Hence alternative means of displaying visual feedback is absolutely necessary. Wang, et al. [48] provided various widget designs that take into consideration the hand occlusion problem.

5.2.4 Gestures

The multitouch interface provides developers an excellent opportunity to work with finger gestures. It is important to note that gestures are only a part of interface design to provide proper inputs and hence gestures alone should not be given more importance than GUI elements, GUI layout or other touch input properties in order to ensure overall efficiency. Until now hand based gestures have not been studied in detail. A hand-finger gesture is executed to convey some useful information. The classification of the gestures was discussed in section 2.2.7. It is clear that the classified gestures are used for human communication. When it comes to human-computer communication, these gestures have their limitations. The major limitation is the absence of 3D gestures since the users are able to utilize only 2D gestures for multitouch input. In the case of multitouch interface, the gestures are classified as direct and symbolic [59].

Direct Gesture

The direct gesture allows user to manipulate the objects directly using single or multiple fingers [33]. Figure 21 shows examples of direct gestures; users can translate rotate or scale an object
using one and two fingers respectively. The object is scaled depending on the distance between the two fingers or rotated depending on the orientation of the fingers.

**Symbolic Gesture**

Symbolic gestures are patterns drawn in order to convey a specific event. These symbols or patterns can be of any shape. For example one can draw a circular pattern to invoke menu or erase an entity. Figure 21 shows different symbolic gestures. For symbolic gestures there must be a gesture recognition engine such as a neural network to identify the gesture and to execute that particular command. This additional step slows down the work flow.

![Figure 21: Examples of direct gesture (A) and symbolic gesture (B) [59][33]](image)

5.3 Multitouch Hardware Research

The above section highlighted the key elements of multitouch software interface that must be considered for developing touch based applications. In Chapter 3 we discussed various types of multitouch hardware, their advantages and disadvantages. We now take a look at the common
issues surrounding the multitouch hardware area when it comes to designing multitouch applications.

The first and the major issue is that of false blob. Users cannot inadvertently touch anywhere on the input screen since it will be considered as a touch event. This is a major problem when the application needs users to type text. The conventional keyboard, as discussed in section 2.2.6, provides tactile feedback and with some practice the users are able to type without looking at the keyboard provided their hands are rested on the home position. In case of multitouch interface one has to incorporate soft keyboards to facilitate text entry. As discussed in section 2.2.6, soft keyboard suffers from non-tactile feedback which prevents users from typing without seeing it. Moreover, in case of multitouch interface, the user has to click one key at a time without resting their palm on the home position (as it triggers false blob) which causes decrease in typing efficiency.

The second issue focuses on the multitouch surface area. The main advantage of indirect input type (like mouse) is that the user can sit in one position and hover the pointer anywhere on the screen. In case of a large multitouch display this is not possible since it is a direct type of input and requires the user to constantly move in order to select the desired entity. Large display areas are usually used for presentation or visualization rather than performing tasks. In case of touch-based interaction the user need large displays for performing tasks because of the size of the hand. Small display interaction with the hand is inconvenient for drawing or manipulating objects.

The third issue which is given less focus is the orientation of the touch surface. As outlined by Forlines, et al [54], large horizontal displays suffer from perspective distortion leading to more
selection errors. However vertical displays do not suffer from such problems. Chapter 6 will discuss more on perspective distortions. For applications involving drawing, modeling, manipulating, both horizontal and vertical displays are not the best option for multitouch interaction. The effects of the multitouch surface inclination are less investigated and the relation between type of applications and orientation is not well established.

The final issue is the type of multitouch hardware being used. Muller [33], from his experiments, pointed out that the performance of the optical-based multitouch depends upon the type of projector used and the speed of the computing system. Hardly there is any literature that compares the different types of multitouch hardware for a specific application. Hence this area is also wide open for investigation; it can yield valuable information which is not available today.

5.4 Computer Aided Design (CAD)

CAD applications which are an important field among other engineering software applications is discussed in this section and the factors which must be considered for incorporating the multitouch interface are highlighted. CAD software is a multibillion dollar industry serving other industries across the world to design and develop products in a cost effective manner. Conceive, design, realize and service form the backbone of product life cycle. The Design stage relies heavily on CAD tools. There are many types of input devices used for modeling in CAD. Input devices such as Mouse, Digitizers, Joystick, and Trackball aid creation, manipulation of 2D and 3D models. The most commonly used input device is mouse. Mouse is used for creation (i.e. points, lines, and curves) and manipulation (i.e. translate, rotate, zoom). The Design stage forms the early stage of a product development lifecycle. This stage is an important part of product development lifecycle since it is very easy and cost effective to make changes and finalize the
product design in this stage. CAD modeling has two main parts: 2D sketching and 3D modeling. CAD designer mostly start off with 2D sketches and convert them to 3D models. Features such as loft, extrude, and sweep are applied to 2D sketches to convert them to 3D models (see Figure 22).

![Illustration of various surface and solid modeling operations in CAD](image)

**Figure 22:** Illustration of various surface and solid modeling operations in CAD

Introducing multitouch interaction for performing CAD operation is not a simple task. There are many factors that affect CAD operations which command their investigation..

### 5.4.1 Selection Task

Many operations in CAD involve the selection of a particular entity. The entity can be a point, an edge, a face or the entire solid object itself. Some operations involve the selection of a point to initiate the process. Let us consider an example as shown in figure 23. The task involves selection of the yellow point. In case of the mouse it is much easier to see and select the point since the mouse pointer almost eliminates the occlusion problem. That is not the case when selecting the point using finger touch (see Figure 23); one cannot see the point they are touching. This is a classic example of an occlusion problem. Moreover, the small dimensions of the point
do not follow the minimum size requirements for the finger touch system as discussed in the section 5.2.2.

Figure 23: Selection of a point using mouse pointer (A) and finger touch (B)

Benko, et al. [53] and Malik [60] have introduced novel and precise selection techniques. Their studies were limited to conventional GUI elements and did not consider very small objects like a point. Further research has to be conducted to solve the dual problem of precise selection and finger occlusion avoidance.

The research study should not focus on selection issues of points, planes and edges in isolation. There are scenarios where the point edges and plane intersect with one another. Consider a case as shown in figure where the user has to select the blue point. It will be an easy task to do with the help of a mouse device, but in case of finger touch (see Figure 24) the point is occluded and there may be chance that the system gets confused whether to select the point, the three intersecting edges or the three intersecting planes. This problem cannot be solved simply by comparing the shortest distance between the centroid of the finger touch and the point, edge and face since there will be a problem if any two shortest distance yields the same value. Such cases must be backed by intelligent and robust algorithms to achieve the task. One may argue that for
selecting an edge or a plane the user can touch at the center of these entities rather than at the intersection. In many CAD operations one needs to work with the intersection points. Hence one needs to have different approaches for finger touch based GUI element selection and selection of any entity within the design space.

![Figure 24: Selection of an intersecting point using mouse pointer (A) and finger touch (B)](image)

5.4.2 Dimension Task

CAD aims at creating models with particular dimensions. During the execution of a specific operation users may have to input dimensions one or multiple times depending on the operation. This again needs one to have a soft keyboard. Though dimensions mainly deal with numbers, the soft key board again gives rise to reduced efficiency as compared to a normal keyboard as discussed in the section 2.2.6. If one designs a system where users can use normal keyboard for input, it introduces the key problem of device acquisition time. One can certainly turn to multi modal input by introducing speech input for dimension. Researchers have explored the
multimodal input by combining pen and voice inputs. This type of combination proved to be very good but still suffer from complementary strengths and weakness [61] [62]. The area of multimodal input for specific CAD tasks holds tremendous potential for future research.

### 5.4.3 Gestures

The key to improve CAD modeling efficiency is to eliminate unwanted steps that come between designers thought process and the final desired result. Consider a very good example of CAD manipulation where one is required to rotate, scale and translate the 3D model. In conventional CAD software one is required to select the rotate icon, then select the 3D model and finish the process. Now in this process one can eliminate the selection of rotate icon by introducing a gesture for rotating the 3D object. This will ensure that unnecessary manipulation and icon selections are eliminated. Selection, creation, modification and manipulation are inherent part of CAD operations. Creation can be further divide into, (1) 2D sketch creation (line, circle, curve, etc.), (2) 3D model creation for both solid and surface modeling (extrude, loft, sweep, etc.). Modification involves tasks such as copy, trim, erase and mirror that are usually used for 2D sketches. Manipulation involves translation, rotation, scaling tasks for both 2D sketches and 3D models. The example given at the start of this section was for 3D manipulation and is also applicable for 2D sketches. Though gestures seem to be a good option for manipulation, the same cannot be said for selection and creation. The best and intuitive way for selection of an entity is to point at it; but as discussed in the section 5.2.3, it has its own issues. One may think of creating gestures for selection such as circling around a point to select it, but this becomes tiresome since selection operations are repetitive tasks. So how about using gestures for creation and modification tasks? Symbolic gesture based sketch modifications are described by Lili, et al.
where the gestures are recognized by a neural network engine and corresponding commands are executed (see figure 25).

![Gesture based modifications](image)

**Figure 25:** Gesture based modifications [63]

In order to obtain full potential of the gesture-based sketching and modification one needs to eliminate different icons for sketching and modifying entities since the goal of having gestures is to remove the intermediate selection steps. In other words there must be free hand drawing system for 2D sketching where the neural network recognizes and creates circle, rectangle, etc. If one uses gesture as shown in figure for sketch modification then the system will get confused and may consider the gestures to be a part of the sketch and not as a command. One of the methods to prevent this is to have icons. Hence there is an inherent tradeoff between realizing the full potential of gestures vs. GUI. This does not imply that gestures should not be used in combination with GUI elements. The right combination of gestures and GUI elements can only be obtained by further research in this area. Current research favors direct gesture over symbolic gestures, but it is also important to consider symbolic gestures for CAD operations. Complex models may be built in a simple method using symbolic gestures. One needs to execute the symbolic gesture and the system creates the model corresponding to the gesture. Theoretically one can design the entire CAD system using symbolic gesture but at a huge cost of user memory.
Only future research can indicate the optimal balance between direct and symbolic gestures for CAD operations.
Chapter 6
Evaluation of Multitouch Interface

6.1 Unimanual Experiment

In this section we will discuss experiments performed by various researchers to evaluate the performance of multitouch interface and compare them with that of the mouse device. Forlines et al. [54] and Muller [33] have performed similar kind of experiments to evaluate the performance of a multitouch device. Forlines conducted experiments for both unimanual and bimanual input system. The importance of these experiments cannot be understated since it helps us understand the issues related to performing task using multitouch interface. The description of the experiment is given in the following section.

![Figure 26: Unimanual experiment [54]](image)

In the unimanual experiment the user is required to place the pointer at home location as shown in Figure 26. A green target and a grey dock will appear. The distance between the target and the dock is same as the distance from the home location to the target. The user is expected to quickly and accurately select the green target and put it on the dock. Docking occur automatically once
the target is dragged within 5 pixels of the center of the dock. A docking error occurs whenever the user releases the target before it is docked. A selection error occurs whenever the user misses the target. A Fitts law model [54] for both mouse and multitouch interface is used for selection portion of the experiment.

5.2 Unimanual Experiment Results

Selection time, docking time, selection error and docking error were observed. The results indicated that the mouse performed better than the touch input device. Forlines, et al. [54] highlighted the factors affecting selection error in touch input device, which are discussed below.

The first factor is the perspective distortion due to large display. The perspective distortion occurs when the distance between user and the display is large. This however can be avoided in vertical display than horizontal display. From Figure 27 one can see that the properties of the finger touch in case of the vertical display remains the same irrespective of the touch position. In case of horizontal display, the finger touch changes when the distance between user and the display increases. The second factor is the physical contact of the finger touch. In case of vertical
touch, see figure, irrespective of the position of the touch, there is less possibility of other fingers coming in contact with the display other than the index finger. In case of horizontal display (see Figure 27) when the distance between display and the user is large then there is high possibility of fingers other than the index finger that can touch the display surface. These factors must be considered while designing the software front end interface.

### 6.3 Bimanual Experiment

The bimanual experiment conducted by Forlines, et al. [54] is described below. The experiment is similar to that of unimanual input (see figure 28). The major difference is that there are two home positions for two cursors. The users have to drag the target to the dock using two index fingers in the case of touch input and in the case of the mouse device, two mice are used. Docking error and selection errors are defined in the same way as that of the unimanual experiment.

**Figure 28: Bimanual experiment [54]**

Forlines, et al [54] concluded from the experiments that the mouse is better suited for unimanual tasks and the multitouch is well suited for bimanual tasks. This is particularly important since designers must exploit full potential of multitouch interface and not treat them as single input
tablets. Similar set of experiments were designed and tested for evaluation of multitouch devices by Muller [33]. Muller [33] experiments involved 1D pointing task, 2D pointing tasks and object manipulation tasks. For detailed description of Muller’s experiments one can refer to Muller [33]. The results of Muller’s [33] experiments are discussed below.

For 1D pointing task experiment Muller [33] used Fitts law to evaluate the performance of the mouse and the multitouch device. The results showed that the mouse had low starting costs and the multitouch had high starting costs. As the index of difficulty increased the multitouch device outperformed the mouse device. Comparing the index of performance of the two devices, it was noted that the multitouch device performed better than the mouse device. This result was true even in case of 2D pointing experiment with multitouch device yielding better index of performance than the mouse.

In case of object manipulation experiments four tasks were designed to be performed by the users. Test A involved translation of an object (dragging of the object is involved). The surprising result was that the selection time was not much effected by target width and distance. The total completion time for task A proved that mouse is better suited for pointing and dragging tasks. Tests B and C involved (translation + scaling) and (translation + rotation) tasks respectively. In case of the multitouch device test B yielded better than test C. The reason for this is the 180 degree rotation incorporated in test C. As the index of difficulty increased, the multitouch outperformed the mouse device in test B and test C. Test D involved translation, scaling and rotation tasks. The results for test D again revealed the better performance of the multitouch device over the mouse device as the index of difficulty increases.
The results from the Muller [33] experiments present an exciting opportunity for other researchers to further probe into the field of multitouch input devices for performing tasks very specific to a particular application. Hence the investigation of the multitouch interface for CAD applications has good research potential.
Chapter 7

Design and Evaluation of 3D CAD Floating Toolbar

In the previous chapter we have identified the possible issues with using multitouch interface for performing CAD operations. We conducted simple mouse emulation experiments to verify the existence and occurrence of such issues.

The objective of the mouse emulation experiments is to identify the factors affecting the task completion time and the error rates when performing the given 3D CAD operation using finger touch as compared to performing the task using a standard mouse. For our experiments we compared the standard mouse with two other finger touch mouse technique. For the first method we considered a drag state finger touch (DSFT) technique and for the second method we considered a track state finger touch (TSFT) technique. In case of DSFT, the finger touch of the user acts as a mouse left click. So when the user touches the screen the left click is activated. Although for this technique one may use any finger, we considered the index finger of the dominant hand as the primary finger touch. Since drag and track state cannot exist simultaneously, the users will not be able to hover the mouse pointer in case of the DSFT. The second method is the TSFT, where the track state is enabled. In this technique, wherever the user places the index finger, the mouse pointer hovers and no event is triggered. The user has to then tap the thumb while keeping the index finger in the same position in order to trigger a mouse left click event (see Figure 29). This technique is similar to the technique used by Matejka et al. [64]. For our experiments, we have considered the left click functionality and ignored middle button and right click functionality.
The main objective of the mouse emulation experiments is to compare the standard mouse with DSFT and TSFT techniques and identify the major issues in the finger touch techniques when it is used for performing 3D CAD operations. The main observations include overall task completion time, errors and error concentration.

7.1 Experiment Setup

For this experiment we used a standard optical mouse as one of the input device. For DSFT and FSFT techniques we use a FTIR based multitouch table. The table dimensions are 60 x 60 x 90 cm. Clear acrylic sheet was used to serve the display surface. X-box 360 live vision camera, modified with a 850 nm infrared bandpass filter was used for capturing the finger touch. 850 nm infrared ribbons were placed along the circumference of the acrylic sheet to serve as a source of infrared light. A white table cloth was used underneath the acrylic sheet to act as a diffuser surface. A long throw EPSON projector was used in combination with a mirror to project the output on to the diffuser surface. A picture of the setup is shown in Figure 30.
7.2 Experimental Procedure

Figure 31 shows the experimental procedure to be followed by the test subjects in the order a-n.

The entire procedure is an operation cycle. It consists of the following sub tasks:

1. Select the rectangle tool and drag the cursor for create a 5 x 5 grid rectangle (a, b)
2. Select the extrude tool and drag the cross section to a height of 5 grid (c, d)
3. Select the circle tool and drag the cursor to form 0.5m radius circle (e, f)
4. Select the extrude tool and drag the cross section to a length of 0.5 m (g, h)
5. Select the erase tool and select the edge of cylinder to erase it (i, j)
6. Select the paint tool, then select any texture form the drop down menu and assign them to the three faces of the cube (k, l)
7. Select the dimension option from the tools menu, then select an edge and drag it in order to create dimension and modify it using soft keypad or normal keyboard (m, n)
Figure 31: Sub tasks associated with the experiment
We used Google Sketchup 7.0 to perform the experiment. The sub tasks were designed with a very specific aim. For tasks $b$ and $d$ grid was provided and for tasks $f$ and $h$ there was no grid. This will help us to observe the effect of precise measurement task in the absence of grid in case of the finger touch technique. When it comes to tasks $j$ and $n$, the edge selection precision of the touch technique is tested. The rest of the sub tasks mainly involves selection of GUI elements and this again will help us understand the nature of finger interaction with the GUI. The tasks are performed once with toolbar at the top position and once with toolbar at the bottom position. Hence for three methods there are two toolbar setting which gives six combinations.

The data was gathered using two methods: desktop activity recorder [65] and Video recording. The desktop activity recorder was used to record all the activities taking place on the screen. The video recording was used to record the output screen and the fingers of the subjects which enabled us to gather the error data. Touchlib mouse driver was used for FSFT and TUIO mouse driver was used for TSFT technique.

All in all 14 subjects were invited for performing the experiments. Four subjects had prior experience in CAD software. All the subjects had high experience in using a computer. Two subjects had moderate level of experience in multitouch interface. Each session took 1.30 hrs. to 2 hrs. Upon their arrival, the subjects we briefed on the nature and the objective of the experiment. They were then given training on how to use the multitouch system for both the techniques. Once the subjects were comfortable with the device the experiments were conducted.
7.3 Experimental Results

When the toolbar position was at the top, the average task completion time was 59.28 sec, 86.5 sec, 90.14 sec for mouse, DSFT and TSFT respectively. For the bottom toolbar position the average task completion time was 59.85 sec, 86.57 sec, 91.71 sec for mouse, DSFT and TSFT respectively. For the toolbar position at the top the average error rate was 0.92, 3, 3.07 for mouse, DSFT and TSFT respectively. For the toolbar position at the bottom the average error rate was 0.64, 2.71, 3.14 for mouse, DSFT and TSFT respectively.

Figure 32: Average completion time and error rate according to toolbar position
Both the average task completion time and the error both yielded similar pattern of results with mouse out performing DSFT and TSFT. By performing t-test we can comment on the statistical differences between the mouse, DSFT and TSFT.

**Paired t-test on Mean Task Completion Time**

Let us consider the average task completion time for the mouse, DSFT and TSFT as $t_m$, $t_{dsft}$, $t_{tsft}$ respectively for the top toolbar position. We wish to compare the null hypothesis with that of the alternate hypothesis.

$$H_0: t_m - t_{dsft} = 0 = \Omega$$

(1)

$H_0 =$ the null hypothesis; $t_m =$ the mean of the task completion time using the mouse; and $t_{dsft} =$ the mean of the task completion time using the drag finger touch technique.

**Test statistic:**

$$T_0 = \frac{(D - \Omega)}{(S_D / n^{1/2})}$$

(2)

The alternate hypothesis is given as follows

$$H_1: t_m - t_{dsft} \neq \Omega \quad (|T_0| > T_{\alpha/2, n-1})$$

(3)

$$H_1: t_m - t_{dsft} > \Omega \quad (T_0 > T_{\alpha/2, n-1})$$

(4)

$$H_1: t_m - t_{dsft} < \Omega \quad (T_0 < - T_{\alpha/2, n-1})$$

(5)
Number of test subjects \( n = 14 \). Hence, degrees of freedom \( n - 1 = 13 \)

Considering value of \( \alpha \) as 0.05, the corresponding value of \( T_{\alpha/2, \, n-1} \) is 2.160

Minitab analysis of paired t-test between the mouse device and DSFT yielded a \( T_0 \) value of -13.91

Given that \( |T_0| > T_{\alpha/2, \, n-1} \) the null hypothesis has to be rejected.

The condition \( T_0 < - T_{\alpha/2, \, n-1} \) is satisfied and hence \( H_1: \ t_m - t_{dsft} < \Omega \) is considered for the direction. This means that statistically the mouse yields lower mean completion time value than the DSFT method.

Minitab analysis of paired t-test between the mouse device and TSFT for mean task completion time yielded a \( T_0 \) value of -11.30

Because \( |T_0| > T_{\alpha/2, \, n-1} \) the null hypothesis has to be rejected.

The condition \( T_0 < - T_{\alpha/2, \, n-1} \) is satisfied and hence \( H_1: \ t_m - t_{dsft} < \Omega \) is considered for the direction. This means that statistically the mouse yields lower mean completion time value than the TSFT method.

Minitab analysis of paired t-test between DSFT and TSFT for mean task completion time yielded a \( T_0 \) value of -1.28

Since the condition for alternate hypothesis is not satisfied, the null hypothesis cannot be rejected. The same results were obtained in the case of toolbar bottom position.
Paired t-test on Mean error rates

In case of errors when toolbar was positioned at the top, we again conduct a paired t-test. The null hypothesis and the alternate hypothesis are given as follows:

\[ H_0: \ e_m - e_{dsft} = 0 = \Omega \]  \hspace{1cm} (6)

\( H_0 \) = the null hypothesis; \( e_m \) = the mean of the error using the mouse; and \( e_{dsft} \) = the mean of the error using the drag finger touch technique.

Test statistic:

\[ T_0 = (D - \Omega) / (S_D / n^{1/2}) \]  \hspace{1cm} (7)

The alternate hypothesis is given as follows

\[ H_1: \ e_m - e_{dsft} \neq \Omega \hspace{1cm} (|T_0| > T_{a/2, n-1}) \]  \hspace{1cm} (8)

\[ H_1: \ e_m - e_{dsft} > \Omega \hspace{1cm} (T_0 > T_{a/2, n-1}) \]  \hspace{1cm} (9)

\[ H_1: \ e_m - e_{dsft} < \Omega \hspace{1cm} (T_0 < - T_{a/2, n-1}) \]  \hspace{1cm} (10)
Number of test subjects \( n = 14 \). Hence, the degree of freedom \( n - 1 = 13 \)

Considering value of \( \alpha \) as 0.05, the corresponding value of \( T_{\alpha/2, n-1} \) is 2.160

Minitab analysis of paired t-test between the mouse device and DSFT for mean errors yielded a \( T_0 \) value of -4.60

Since \( |T_0| > T_{\alpha/2, n-1} \) the null hypothesis has to be rejected.

The condition \( T_0 < - T_{\alpha/2, n-1} \) is satisfied and hence \( H_1: e_m - e_{dsft} < \Omega \) is considered for the direction. This means that statistically the mouse yields lower mean error than the DSFT method.

Minitab analysis of paired t-test between the mouse device and TSFT for mean errors yielded a \( T_0 \) value of -7.81

Given that \( |T_0| > T_{\alpha/2, n-1} \) the null hypothesis has to be rejected.

The condition \( T_0 < - T_{\alpha/2, n-1} \) is satisfied and hence \( H_1: e_m - e_{dsft} < \Omega \) is considered for the direction. This means that statistically the mouse yields lower mean error than the DSFT method.

Minitab analysis of paired t-test between TSFT and DSFT for mean errors yielded a \( T_0 \) value of -0.20

Since the condition for alternate hypothesis is not satisfied, the null hypothesis cannot be rejected. The same results were observed in the case of toolbar bottom position.
7.4 Discussion

Task Completion Time

The task completion time for mouse was significantly less than that of DSFT and TSFT. A significant difference is not reflected when comparing the completion time of the DSFT with that of the TSFT method. To understand the reason one has to dive into the subtasks performed. The subtask b required the subjects to drag the index finger in order to draw the square. This dragging subtask was more time consuming in DSFT and TSFT method as compared to the mouse. The primary reason for this slowdown is the friction between the finger and the display area. This phenomenon is also confirmed by experiments conducted by Matejka et al. [64].

The desired target of the square was 5 x 5 grid. No numbers were allocated to the grids and hence the subjects had to visually calculate the grid dimensions. We believe that by providing grid numbers as a visual feedback one may reduce the time taken for drawing geometries by finger dragging process. With further research in new materials that are compatible with optical based multitouch table, one may arrive at an optimal solution.

The dimensioning subtask n took the maximum time which affected overall completion time. The DSFT and the TSFT method yielded large values of completion time as compared to that of the mouse device. Our observations showed that the time difference was double in case of DSFT and TSFT method as compared to the mouse device. The subtask n required selecting, dragging an edge and changing the dimension value using a soft keyboard. Though one was required to enter only numbers, the problems associated with soft keyboards surfaced as discussed in section 2.2.6. Typing one number at a time reduces the efficiency. Moreover the problem of tactile feedback and typing without seeing the screen still exists with the soft keyboard. In CAD
applications one cannot escape the dimensioning step. The developers can minimize the dimensioning steps with the help of grid for drawing 2D primitives and ruler for operations such as extrude. It is a good practice to provide dynamic visual feedback with grids and rulers. The same observations on task completion time were recorded in the case of drawing circle and extruding circle subtask. In these subtasks the subjects had to control the circle dimension and the extrude length to 0.50m. The subjects were not provided grid for these subtasks. The dragging friction slowed down the process. The other important factor for slowdown was that the subjects were asked to perform the subtask with precision measurement of 0.5m. The subjects found it difficult to control the precision measurement using finger touch. This problem was magnified when the subjects reached 0.49m or 0.51m after which they had to bring the measurement to 0.50m. At this stage the subjects were trying to adjust the measurement by not moving finger but by rocking the finger at 0.51 or 0.49. The system could not recognize the minute rocking and it required the user to move the position of the finger causing the measurement to overshoot on either side on 0.50m. Precise control over the geometries is absolutely essential for CAD applications. This is particularly important when one needs to reduce the dimension step and introduce grids and rulers.
Error Concentration

From Figure 32 one may see that there is not much difference between mean error rates of DSFT and TSFT methods. However it is important to discuss the error concentration of DSFT and TSFT corresponding to the subtasks performed.

Figure 33: Error concentration for different subtasks: toolbar top position (top graph), toolbar bottom position (bottom graph)
The error concentration for both toolbar position involving DSFT and TSFT methods are displayed in Figure 33.

**Common Error Zone for DSFT and TSFT Methods**

The draw circle \( f \) and the extrude circle \( h \) subtasks yielded most of the errors in case of DSFT and TSFT methods. The toolbar position did not have a significant effect on the error rate for these subtasks. The reason for this is already discussed in the task completion time section. Precise control of measurements was identified as the main reason. But in case of TSFT, the subjects had to control the dimensions with two fingers as compared to DSFT method where the subjects used single finger. This led to more error rates in TSFT method as compared to DSFT method for subtasks \( f \) and \( h \).

**Selection Errors**

Overall the selection errors for icons and menus were less. This is due to the fact that the icons were 10 x 10 mm in dimensions which could accommodate good finger selection precision. One area to concentrate for selection errors in the design space, mainly the selection of edges for subtasks \( j \) and \( n \). The subtasks \( j \) and \( n \) required the subjects to select an edge for deletion and dimension respectively. For these subtasks (erase edge, edge selection) TSFT method yielded low error values than the DSFT method. The main reason lies in the state of the techniques. The DSFT method incorporates drag state and the track state is absent. This means that the subjects were not able to hover the pointer over the desired edge. The subjects made multiple attempts to select the edge and thus leading to higher error values. The TSFT method allowed track state which allowed the subjects to hover the pointer above the edge and select it with least amount of attempts. Though the completion time for these subtasks did not differ significantly for both
DSFT and TSFT methods, the error values were significantly different. This poses a classic problem of device states. The majority of the input devices support either tracking or dragging states but not both. Hinckley et al. [66] designed a stylus and tablet based system which incorporated both dragging and tracking states. This system enabled users to hover the stylus just above the screen (not making contact) in order to hover the mouse pointer. The moment the stylus touched the screen the drag state was enabled. This becomes easy in case of stylus since the tablet can sense the stylus proximity to the screen. In case of currently available multitouch systems, the proximity of the fingers cannot be judged and hence both the states cannot be incorporated. Among the optical-based multitouch systems only RI systems can be modified to detect finger hovering. Since the infrared distribution in RI system is uneven, the continuity of the detection of the fingers cannot be guaranteed. In future more multitouch hardware research may yield good solutions to enable both drag and track state. This can be beneficial for CAD systems for reducing selection errors.

7.5 Proposed Floating Toolbar for 3D CAD Operations

Based on our findings, we have introduced a floating toolbar concept. Using this concept one can use a finger gesture to perform different operations. The toolbar consists of alphabets context buttons (E=Extrude, -E=Subtract Extrude, R=Revolve, -R=Subtract Revolve, S=Sweep, -S=Subtract Revolve, L=Loft). The example shown in Figure 34 describes how to perform an extrude gesture. The user has to keep the index finger of one hand on E and drag the cross section using the index finger of the other hand. We call this a drag gesture. For sweep, loft, revolve gestures the user has to keep the static finger on the respective operation button.
The advantage of the floating toolbar is that it can be placed anywhere on the screen to perform the drag gesture according to the convenience of their dominant hand. A right hand dominant person will usually use the right hand index finger to drag and the left hand index finger to indicate the static position. We have limited the number of operations in the floating toolbar for the ease of performing experiments. The floating toolbar can be enabled to include all modeling operations and thus help designers to access all operations using single floating toolbar and execute those operations using drag gesture. The toolbar is scalable and rotatable so that the users can adjust to their finger sizes. The icons have minimum dimensions (10mm x 10mm) to ensure the selection precision. The icon size also adjusts to the scale factor of the toolbar thus enabling the users to have larger icons according to their comfort. Though we identified finger drag friction as a major problem while executing drag gestures in our experiments, we believe

Figure 34: Proposed floating toolbar for 3D CAD operations
that for gesture based 3D CAD operations one cannot neglect the drag gesture since it is an intuitive gesture.

7.6 Proposed Evaluation Technique for Gesture Based CAD Operations

We recognized a need for different evaluation parameters when it comes to gesture based interaction for CAD operations. Federico [67] and Nielsen et al. [68] proposed that gesture and the user interfaces must be evaluated based on learnability, memorability, error rates, efficiency and accuracy. Usually learnability, memorability is got from user reviews which are rated on a scale of 1 to 10. In our experiments we measured completion time and error. The factors affecting the errors were also discussed. In spite of this, our experiments did not involve different gestures. Though for TSFT one may consider the left tap as a gesture, it still mapped the mouse functionality which is repetitive. Hence, how to evaluate a CAD operation if it incorporates different gesture is a challenging question. Our proposed evaluation technique allows researchers to quantify these parameters (learnability, memorability, efficiency) in terms of errors which of course are measurable. Figure 35 gives the definition of an operation cycle and sub-operation. Based on these definitions the proposed evaluation parameters are described as follows:

![Figure 35: Schematic diagram of an operation cycle and sub-task](image-url)
Memorability

\[ M_{(sub-operation)} = \frac{\sum_{i=1}^{n} O_i}{n} \]  

(11)

\( M_{(sub-operation)} \) = Memorability value for a sub-operation; \( O \) = Number of times a sub-operation is repeated until it is performed without an error; and \( n \) = Number of users

The memorability values must be low for good gesture based interface. It is important to note that the memorability values are for subtasks.

Learnability

\[ L_{(operation)} = \frac{\sum_{i=1}^{n} O_i}{n} \]  

(12)

\( L_{(operation)} \) = Learnability value for an operation; \( O \) = Number of times an operation cycle is repeated until it is performed without an error; and \( n \) = Number of users

At first glance the learnability may seem like summation of memorability values. The learnability values are significant and cannot be avoided. The significance is more related to cognitive science. Though the evaluation may yield low memorability for the subtask, it is not necessary that when all the subtasks are put together to form an entire operation cycle the user will perform with same efficiency. In CAD an operation can be performed in many ways. So if there is a gesture-based modeling environment, each operation can have many subtasks with their corresponding gestures. The user may choose any method to start the operation cycle. In
spite of good memorability values, the order in which subtasks are performed may affect the learnability values. So it is important that the gesture design must yield a low memorability and learnability values.

**Efficiency**

\[
E_{\text{(operation)}} = \frac{\sum_{i=1}^{n} T_i}{n} \times 100 \tag{13}
\]

\(E_{\text{(operation)}}\) = Efficiency value for an operation; \(T\) = Time taken to perform an operation; and \(n\) = Number of users

**Accuracy**

\[
A_{\text{(operation)}} = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} 100X_{ij}}{(n)(m)} \tag{14}
\]

\(A_{\text{(operation)}}\) = Accuracy value for an operation; \(X_{ij} = 0\), if an error is generated; \(X_{ij} = 1\), if an error is not generated when the \(j_{\text{th}}\) user performs the \(i_{\text{th}}\) operation; \(n\) = Number of times the operation cycle is repeated; and \(m\) = Number of users

Another important point to remember is that the learnability, memorability and the efficiency must be tested for short term usage and long term usage. The effects of short term and long term usage of the multitouch environment will provide us with the learning curve for that system. This will be helpful for developers to assess the re-training costs which are involved as a result of incorporating new gestures into the system.
7.7 Conclusion and Future Work

We conducted mouse emulation experiments and identified the problems associated with multitouch interaction for CAD operations. Our experiments indicated that the drag state and track state interaction techniques did not differ much in task completion time and error rates. The mouse device performed better than the DSFT and TSFT methods. Both DSFT and TSFT methods yielded high error rates for tasks involving precise control of measurement. TSFT method is more suited for edge selection tasks than the DSFT method. In spite of increased completion time, the incorporation of grids helped reducing measurement errors. Significance of grids and rulers with dynamic visual feedback of the dimension cannot be overlooked.

The research in the field of multitouch interaction for CAD operations is at its infancy. Many problems are yet to be addressed: (1) finger occlusion, (2) selection precision, (3) incorporating both drag and track state, (4) reducing surface friction, (5) hardware research involving display surface inclination, display surface size, false blobs, (6) GUI layout designs, and (7) direct gestures and symbolic gestures.

Form the questionnaire data, 10 out of 14 subjects preferred the mouse device for CAD modeling; 4 subjects preferred DSFT method to that of TSFT and the mouse device; 12 out of 14 subjects said that they will be interested to see a multitouch enabled environment for CAD modeling. One subject reported finger fatigue while performing experiments with the TSFT method. The questionnaire yielded positive response for a multitouch enabled CAD environment. This encouragement supports the case for further research in this area. In future we will evaluate the proposed toolbar and the proposed evaluation technique for gestures based
CAD operations. We will further extend the research in the multitouch hardware domain to address the issues outlined form the current work.
References


34. Harry van der Veen, “FTIR multi-touch display how-to guide (alpha version 0.2)”, 2007.


37. Touch screen table comparison: http://www.touchscreenmagazine.nl/multitouch-techniques/comparisontable/

38. Touchlib: http://www.nuigroup.com/touchlib/


49. Buxton, W., “Multi-touch systems that I have known and loved”: http://www.billbuxton.com/multitouchOverview.html.


67. Marcello Federico, “Usability evaluation of a spoken data-entry interface”.