UNIVERSITEIT VAN AMSTERDAM

Multi-touch displays: design, applications and performance evaluation

L.Y.L. Muller

20th June 2008

Student number: 0604046
E-mail address: lmuller@multigesture.net

Supervisor: Dr. R.G. Belleman

GRID COMPUTING — MASTER'S THESIS

Multi-touch displays: design, applications and performance evaluation

Master's Thesis (*Afstudeerscriptie*)

written by

L.Y.L. Muller

under the supervision of **Dr. R.G. Belleman**, and submitted in partial fulfilment of the requirements for the degree of

MSc in Grid Computing

at the *Universiteit van Amsterdam.*

Date of the public defence:	**Members of the Thesis Committee:**
2nd July 2008	Dr. R.G. Belleman
	Dr. P. van Emde Boas
	Prof. C.R. Jesshope, PhD
	Dr. A. Ponse

UNIVERSITEIT VAN AMSTERDAM

Abstract

When interacting with a regular desktop computer, indirect devices such as a mouse or keyboard are used to control the computer. Results of the interaction are displayed on a monitor. Current operating systems are restricted to one pointing device. With the introduction of multi-touch, a new form of human computer interaction is introduced. Multi-touch combines display technology with sensors which are capable of tracking multiple points of input. The idea is that this would allow users to interact with the computer in a natural way.

Due to recent innovations multi-touch technology has become affordable. For this project a camera based multi-touch device has been constructed at the *Scientific Visualization and Virtual Reality group* of the *Section Computational Science* of the *Universiteit van Amsterdam*. To perform multi-touch point tracking we used Touchlib, a free open source multi-touch framework.

To demonstrate the possibilities of multi-touch input technology we created new rich media applications which are controlled by gestures. Existing software can benefit from multi-touch technology, this is demonstrated in a real-time fluid simulation model and the geographical application NASA World Wind. Multi-touch systems are often stand alone systems that do not have external input devices attached. In order to simplify common tasks, a gesture recognition engine has been designed (Gesturelib).

Through a set of experiments we evaluate how multi-touch input performs on tasks compared to conventional mouse input. Unlike interaction on a desktop computer multi-touch allows multiple users to interact with the same devices at the same time. We present measurements that show how collaboration on a multi-touch table can improve the performance for specific tasks.

Contents

Preface

In this preface, I would like to explain how the thesis subject was chosen. Since high school I have had an interest in the concept of interactive graphics, though at that point the internet was in its early stages and virtual reality was more a science fiction concept than an actively developed topic. After following the master course *Scientific Visualization and Virtual Reality* I knew that I wanted to do a subject related to virtual reality. After a personal appointment with Robert Belleman, he showed a few subjects which I could take as a master project. The subjects were related to computing with graphics hardware (GPU), augmented reality, tiled panel displays and programming applications for the virtual reality environment the CAVE. Unfortunately the GPU project was not available by the time I had to make a decision, the other proposals seemed less attractive. While discussing the proposals, Robert showed a paper of Jeff Han explaining multi-touch sensing using FTIR and the multi-touch open source project Touchlib. A few weeks later Jeff Han was featured on TED Talks (Technology, Entertainment, Design conference) in which he explains the technique and capabilities of multi-touch devices.

During the same period of my thesis subject proposals, NEMO Science Center asked the *Scientific Visualization and Virtual Reality group* of the *Section Computational Science* of the *Universiteit van Amsterdam* if it was possible to develop a virtual reality device for public display. However, the problem with virtual reality devices is that it would only allow a few people to experience virtual reality at the same time. In order to see stereoscopic images special goggles are required which makes the technique less attractive and kids proof. While multi-touch can not be considered as virtual reality, it could provide a platform to get children interest in science and technology.

We decided to combine the multi-touch research project to fit the requirements for NEMO. For this project I would do research on the performance of interaction and collaboration on multi-touch panel displays and after completing the experiments, the panel would be moved to NEMO running science related applications.

Acknowledgements

This thesis would not be possible without the help of many people. First of all, I would like to thank Robert Belleman for suggesting and supervising this project. Jaap Kaandorp and Alban Ponse for their support during the master course. Paul Melis for his support during the entire project. Elena Zudilova-Seinstra for helping out with questionnaire and giving feedback on the experiments. Edwin Steffens for the hardware support. Michael Scarpa for his help during the construction of the first multi-touch prototype (sorry for the broken mirror, that leaves us with 6 years of bad luck).

I would also like to thank a few persons from the NUI group community. David Wallin for creating Touchlib and explaining how to use it. Christian Moore and Seth Sandler for their support on using Adobe Flash. Harry van der Veen for the nice conversation about multi-touch and introducing me to SOCOamsterdam. Sebastian Hartman for explaining the Touchlib tracker system. Johannes Hirche for his support on getting Touchlib working on Linux and Pawel Solyga.

My fellow master students who helped me during the master course, and all persons who helped me out by doing the experiments on the multi-touch table.

SOCOamsterdam director Peter Distol for sharing experiences on building multi-touch tables and for demonstrating Multi-touch Media Application (described in Section 4.3.4) on national television on “In de ban van ’t ding” (VPRO). Ralph Das for sharing ideas on multi-touch application concepts.

NEMO Science Center for funding the project and making my applications available to the general public.

Finally I would thank my parents and my brother for their support and patience.

CHAPTER 1

Introduction

Multi-touch displays are interactive graphics devices that combine camera and tactile technologies for direct on-screen manipulation. Multi-touch technology is not entirely new, since the 1970s it has been available in different forms. Due to the improvement of processing power of modern desktop computers, multi-touch technology does no longer require expensive equipment.

Modern computer interaction consists of a monitor, keyboard and a mouse. Limited by the operating system, it allows us only to control a single pointer on the screen. With multi-touch, multiple input pointers are introduced to the system which all can be controlled independently. Depending on the size of the display multi-touch allows multiple persons to interact with the same display at the same time.

1.1 Existing multi-touch technologies

Since the 1970s several research groups have done research on (multi-)touch sensitive surfaces. Multiple patents [9, 20, 23, 25, 41] demonstrate how camera/sensor based touch sensitive surfaces can be constructed.

In 1997 Matsushita et al. presented the HoloWall [24]. The HoloWall is a finger, hand, body and object sensitive wall. Multiple users can interact with the surface on the front side of the wall. The wall is made out of glass with a rear projection material attached. Behind the wall a digital projector is placed for display. On the same side as the digital projector a camera and an infrared illuminator is placed. When a user or object touches the wall, it reflects infrared light which is captured by the camera.

The Mitsubishi DiamondTouch table [8] is a multi-touch and multi-user sensitive input device. The table works by transmitting an electrical signal to the array of rows and columns embedded in the surface. The DiamondTouch can be used by four persons at the same time. Each user sits on a chair which has a receiver unit attached. The user makes physical contact with the receiver unit. When a user touches the surface, a small current will flows between the array of rows and columns and the receiver unit (through the user's body). The receiver unit can now determine which parts are being touched by which person. The DiamondTouch uses a projector that projects the display on top of the table. The system allows users to control applications with gestures [47]. It is capable of detecting gestures based on touch and gestures based on the shape of the hand.

The SmartSkin [29] is a multi-touch sensitive input device. The sensitive layer consists out of grid-shaped transmitter and receiver electrodes (copper wires). The vertical wires are the transmitter electrodes, the horizontal wires are the receiver electrodes. At predefined intervals a wave signal is transmitted to the transmitter electrodes (only one transmitter per interval). The receiver will receive the wave because it acts as a (weak) capacitor. When the grid is touched by a finger, the signals wave becomes weaker. By continuously measuring the signal strength it is possible to detect touch. The system uses top projection for the display. When using a high density grid, the system is capable of detecting shapes of objects.

In 2005 the reacTable [1, 19] was introduced. The reacTable is a collaborative electronic music instrument with a table top tangible multi-touch interface. The table is capable of tracking fiducial markers (see Figure 1.1) which allows users to add instruments and control them by rotating the fiducial markers. By moving and combining multiple markers instruments can be combined to create unique sounds and music. The table uses infrared light emitters to illuminate the surface. When the user adds an object with a marker on the table, it will reflect the pattern to the camera which can be processed by the computer. The latest version also allows finger multi-touch input.

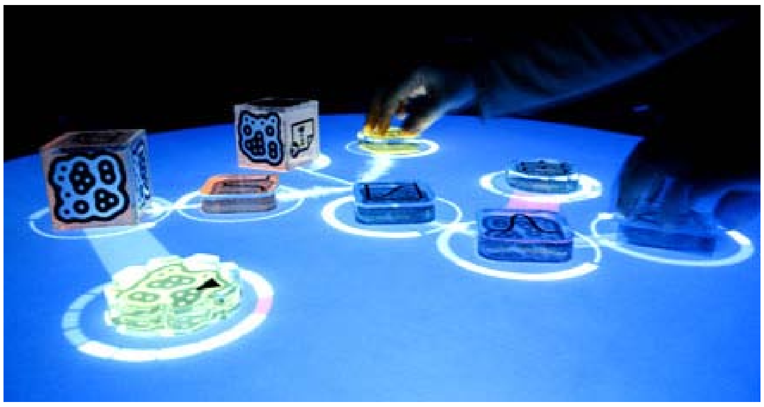


Figure 1.1: Patterns called fiducial markers are attached to physical objects that are recognized and tracked by the reacTable (image taken from [31]).

The JazzMutant Lemur [18] is a standalone multi-touch device designed to control audio and media applications. Because all controls (knobs and sliders) are virtually created on the display, it allows the user to customize the interface to the user's needs. The Lemur communicates over the built-in ethernet interface.

Wilson's [44] Touchlight is a camera based input device which allows gesture based interaction. The system uses a vertical surface made out of acrylic which has rear-projection material attached. Behind the surface a projector is placed for display. The projection material of the Touchlight is more translucent than the one used on the HoloWall. In order to eliminate the background, the system uses two cameras and one infrared illuminator to sense interaction. When a hand reaches near the surface, the hand will be reflecting infrared light. The cameras are placed on different angles, one on the left and one on the right. Due to their positioning, it is required to correct the image for lens and perspective distortion. When merging both images into one, objects near the screen will merge (amplify) and objects further away from the screen will become less visible. Unlike the previously described input devices, this system acts more as a full hand gesture based interaction device rather than a (multi-)point input device. The system has a microphone built in which can detect vibration of the screen. This allows users to interact with the system by tapping the display.

Another device created by Wilson [45] is the PlayAnywhere input device. The PlayAnywhere is a top projection camera based input device which allows multi-touch user interaction. The system uses a short-throw digital projector for top projection, an infrared illuminator to illuminate the environment and a camera to view the desk from above. When a user interacts with the surface, his hands will become illuminated. In order to perform finger tracking, the contour of the illuminated hand is analyzed. Because the hand is illuminated from above, it leaves a shadow depending on the distance between the surface and the hand or fingers. To perform touch detection, the contour of the fingertip is compared with the results of the fingertip shadow. If near the fingertip a shadow is present, the system will detect this as hover. If no shadow is present, the system will detect it as a touch. The benefit of this system is that it allows any flat surface to transform into a multi-touch input system.

In 2005 Han [13, 14] presented a low cost camera based multi-touch sensing technique. The system uses the technique called Frustrated Total Internal Reflection (FTIR, described in more detail in Section 2.2.1) which involves trapping infrared light in a sheet of acrylic. When touching the sheet, the touched spot will frustrate the trapped light causing it to leak out of the sheet. When pointing a camera at the sheet it is possible to view where infrared light is “leaking” and recognize touch.

Early 2007 Apple presented a new cellular phone, the Apple iPhone. Where other touch based cellular phones only allow single point interaction, the iPhone uses multi-touch technology to control the phone. The iPhone senses touch by using electrical fields. On touch, the electrical field will change value which is measured. This allows the iPhone to detect which part of the phone is touched.

Mid 2007 Microsoft presented their version of a multi-touch table, MS Surface [7]. The table looks like a coffee table with an interactive surface. The technique used in the table is similar to the HoloWall. A diffuser is attached to the screen material. The table is illuminated with infrared light from the back. When a user touches the table, it will reflect infrared light which is captured by the cameras inside the table. Because of the use of multiple cameras, the input resolution is high enough to detect objects. Microsoft has demonstrated how fiducial markers can be used to tag objects and allow specific interaction with the table.

A few months after the press release of MS Surface, Microsoft Research Cambridge [17] demonstrated a multi-touch system on a notebook LCD panel. The construction of ThinSight consists of an array of retro-reflective optosensors which are placed behind the LCD panel. Each sensor contains an infrared light emitter and an infrared light detector. When an object touches the LCD panel, it will reflect infrared light which the detector can notice. Because this technique relies on reflecting infrared light, the system allows interaction with fiducial markers.

1.2 Previous studies on the performance of (multi-)touch input techniques

In 1985 Buxton [5] published a paper on touch sensitive input devices. He points out important issues when using touch technology. Buxton describes a three-state model for input devices. In state 0 no physical contact is made with the input device, in state 1 the input device is tracking (movement of mouse cursor) and in state 2 the input device is dragging (button pressed). As an example, Buxton describes a rubber-band line drawing task with a one button mouse and a touch tablet. When using the mouse, a user starts with state 0. After grabbing the mouse and moving the mouse cursor to a proper position it changes to state 1. By pressing the button and moving the mouse the state changes to 2. When the user completes the task, we revert to state 1 and finally state 0 (releasing contact with mouse). When performing the same task on a touch tablet it is only possible to perform touch or no touch. Because only two states are available it is only possible to bind one state (1 or 2) to the touch. When state 1 is bound to touch it is only possible to set the mouse cursor on a specific position. If state 2 is bound, touching the device will cause immediate selection. Buxton suggests that this problem can be solved if the used touch table would be able to sense different levels of pressure. Another problem when using touch tablets is the friction between the user's finger and tablet surface. If the user is required to perform tasks that require long motions with high pressure, this causes inconvenience to the user.

Early research on bimanual input has been done by Buxton [4, 43]. His experiments focused on the performance of two-handed input devices. The experiments included tasks such as object manipulation and selecting lines in large text documents. During the experiments Buxton compared the amount of parallel activity with the time needed to complete the task. Results proved that the efficiency of subject performance is related to the degree of used parallelism. Test subjects had no difficulty in performing the task using both hands.

In 2007 Forlines et al. [12] compared the performance of direct-touch and mouse input on table top displays. The experiments included a two dimensional Fitts Law test and an object manipulation test. For this test a Mitsubishi DiamondTouch was used for direct-touch and two mouse devices were used to perform bimanual object manipulation with a mouse. The results of the experiment show that users benefit from direct-touch when performing bimanual tasks. The results of the mouse input show that it only performs better on tabletop task which required single point interaction.

Terrenghi et al. [34] compared the performance of completing a task in a physical environment and a virtual environment on a multi-touch table. The test participants in the experiment were asked to complete an experiment in which a puzzle had to be solved and an experiment in which photos had to be sorted in three different groups. Results showed that users are faster at solving puzzles in a physical environment, but performance in the sorting photos task increased on the multi-touch table. In overall it was faster to complete both tests in a physical environment than in a digital one. While the users performed the tasks, the amount of one handed input and bimanual input was recorded. Results show that users are more likely to perform bimanual input when touching physical object than using bimanual input on touch screen devices.

1.3 Overview of this work

Recent publications have shown that multi-touch displays have a potential to revolutionize human-computer interaction in the way that they allow intuitive interaction with applications through direct manipulation of graphical constructs. Several research groups have demonstrated that the performance of simple tasks through multi-touch displays show great improvement over conventional methods. Unfortunately, most interactive graphical applications that exist today are not able to exploit this potential because they are designed for a single pointer device such as the mouse. This research project has been divided into four parts:

1.3.1 The design and construction of a camera based multi-touch device

Currently no camera based multi-touch devices are available on the market. In order to perform the required experiments it is necessarily to design and construct a multi-touch device from scratch. The project materials are funded by NEMO (science center) and the *Scientific Visualization and Virtual Reality group* of the *Section Computational Science* of the *Universiteit van Amsterdam.*

1.3.2 The design and implementation of a gestural interaction library

With the introduction of multi-touch, gesture based interaction has become a new standard in the interaction with applications. In order to use gestures on a multi-touch device it is required to design and implement a multi-touch gestural interaction library. The library will retrieve information from the used multi-touch tracking software and process them into gestural primitives. Detected gestures are exposed to application developers through an easy to use application programming interface (API).

1.3.3 Implementation of test case applications

In order to demonstrate the input capabilities and benefits of a multi-touch device a set of (example) applications will be developed. The applications should show examples of collaborative tasks and gesture based interaction on the device.

1.3.4 Performance evaluation of multi-touch enabled tasks

To compare the performance of the multi-touch input device in the field of human-computer interaction a set of tasks will be designed and developed. The tasks focus on point and selection tasks (as seen in regular desktop computing), gesture based interaction and collaborative problem solving.

1.4 Overview of this thesis

This thesis focuses on the construction, capabilities and performance of a multi-touch input device in comparison with traditional desktop computing. Chapter 2 gives an introduction into the currently available camera based multi-touch techniques and design considerations based on our experiences. Chapter 3 explains the software part of the multi-touch system. This includes image processing, blob tracking and gesture recognition. Chapter 4 demonstrates a few of the multi-touch applications that have been developed during this project. Chapter 5 describes the performance evaluation experiments and its results. Chapter 6 contains the result discussion and the final chapter (7) includes the conclusion and future work.

CHAPTER 2

The design and construction of a multi-touch device

For this research project two multi-touch input devices have been built utilizes different techniques. Both multi-touch detection techniques use a camera. The selection of which technique to use depends on the type of environment in which the device will be used and on the type of applications it will be running. This chapter will give a brief explanation on how these techniques work, which design considerations we took and the problems we faced during construction.

2.1 Design considerations

Depending on the target audience a multi-touch panel technique can be selected. Due to collaboration between the NEMO Science Center and the Universiteit van Amsterdam shared design considerations had to be made. According to the collaboration agreement the multi-touch panel would first be constructed at the UvA and used for this thesis research project. After finishing these tasks the multi-touch panel will be displayed to the public at the NEMO Science Center.

The NEMO Science Center had the following requirements for an interactive graphics device:

- NEMO encourages the audience to participate in experiments, the system should be attractive.
- The system needs to be suitable for an audience from 7 up to 70 years.
- The system should encourage users to playing together.
- The system needs to be a standalone device.
- The hardware needs to be 'child proof', which means it is robust and easy to use.

The requirements based on the thesis subject were to construct a reliable multi-touch panel which would allow multiple users to collaboratively solve tasks.

Based on these requirements it was decided to create a table based horizontal multi-touch panel. A table shaped system would allow multiple persons to interact with the multi-touch panel in a familiar way.

2.2 Camera based multi-touch techniques

Camera based multi-touch devices share the same concept of processing and filtering captured images on patterns. In general the interaction can be described as the pipeline in Figure 2.1. The pipeline begins when the user views the scene on the panel and decides to interact. To interact with the device, the user touches the device panel. On the hardware level the camera registers touch. Because the captured frame might not only include the contact points but also a (static) background, it is required to perform image processing on each captured frame. The captured frame is converted to a gray scale image and the static background is removed by subtracting the current frame with a reference frame. As a result the frame only shows white contours (blobs) which are the points of contact. By tracking these blobs the system becomes capable of sensing touch. In order to track the blobs, the positions are transmitted to the blob tracker which matches blobs from earlier frames with the current frame. After processing, events will be triggered which can be used in a multi-touch capable application. The application modifies objects in the current scene according to the new blob positions. The result is returned to the user through the display.

The performance of a camera based multi-touch device depends on the used hardware and software. When a user touches a multi-touch device, it expects the device to respond directly. The responsiveness of the device depends on the time it needs to process the user's input and present the users a result through the display. In the interaction pipeline two levels are important, the hardware and the software level. Using a camera which is capable of 30 frames per second allows smooth interaction with the system. However, this requires a system that can handle image processing and blob tracking in $\frac{1}{30}$ of a second. A combination of smart algorithms implemented in software and fast hardware helps to minimized the latency and increase the responsiveness of the device.

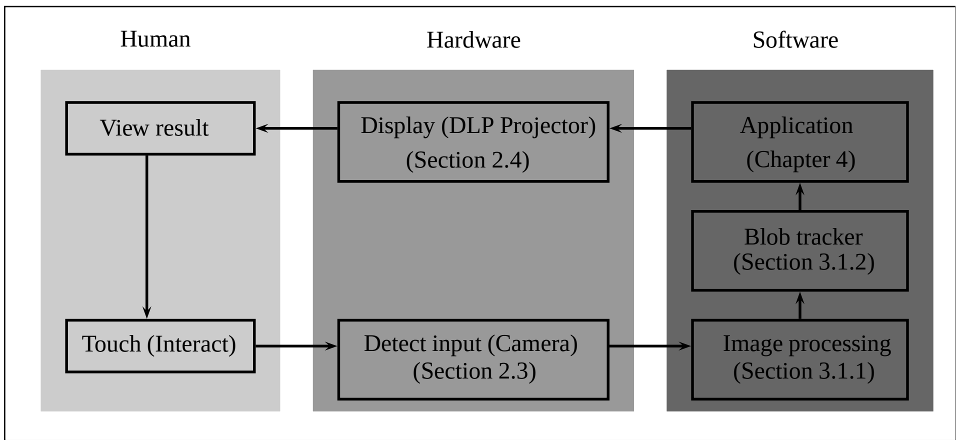

Human
Hardware
Software
View result
Display (DLP Projector) (Section 2.4)
Application (Chapter 4)
Blob tracker (Section 3.1.2)
Touch (Interact)
Detect input (Camera) (Section 2.3)
Image processing (Section 3.1.1)

Figure 2.1: A graphical overview of the multi-touch interaction pipeline. The indicated sections contain additional information on each part.

2.2.1 Frustrated Total Internal Reflection

Our first prototype [26] was based on the technique called Frustrated Total Internal Reflection (FTIR) which has been presented by Han [13]. Thin plates of translucent materials such as acrylic have the property of trapping light by reflecting it continuously on the edges.

> *"Total Internal Reflection (TIR) is an optical phenomenon that occurs when a ray of light strikes a medium boundary at an angle larger than the 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, so effectively all of the light is reflected. The critical angle is the angle of incidence above which the total internal reflection occurs."* - Wikipedia [42]

Infrared light is being used because it is invisible to the human eye but detectable by digital cameras. In the prototype infrared LEDs were mounted on four sides of the sheet of acrylic. When the infrared LEDs are turned on, infrared light will become trapped in the sheet of acrylic. By touching the acrylic the TIR will be frustrated (because of a changing refractive index) causing infrared light to leak out on the touched spot. The fingertip will reflect the infrared light to the camera.

Figure 2.2 shows a side view of the construction. Depending on the angle on which the LEDs emit infrared light, additional baffles are required to prevent 'leaking' infrared light reaching the camera. On the rear side (bottom) of the display a

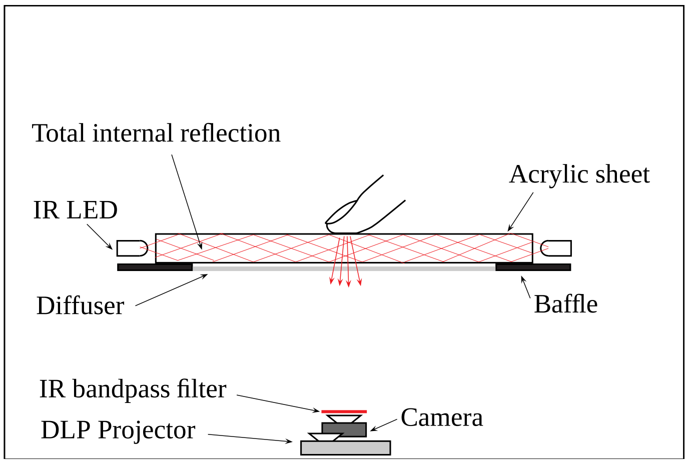


Figure 2.2: Schematic view of the multi-touch panel using frustrated total internal reflection.

diffuser material is placed for two reasons. First it prevents the camera from seeing objects behind the screen and thus providing a cleaner stable input signal. Secondly it functions as a projection screen for the DLP projector. Because touching the display would frustrate the TIR, there is a small gap between the diffuser and the display (a few mm). With this basic setup the performance mainly depends on how 'greasy' the fingertips of the user are. Wet fingers are able to make better contact with the surface. Dry fingers and objects will not be able to frustrate the TIR.

To overcome this issue it is recommended to add a 'compliant layer' (see Figure 2.3). Instead of frustrating the TIR by touching the acrylic directly, a compliant layer is used which acts as a proxy. The compliant layer can be made out of a thin (1-2 mm) silicon material such as ELASTOSIL©M 4641 as suggested by van der Veen [37]. On top of the compliant layer a rear projection material such as *Rosco Gray #02105* is placed. The projection material prevents the compliant layer of becoming damaged and also functions as a projection screen. Additionally, this has the advantage that the fingers touch the projected images directly. With this setup it is no longer required to have a diffuser on the rear side.

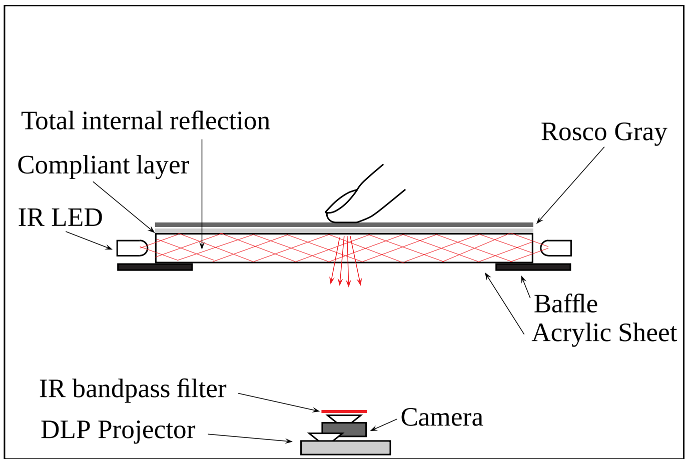


Figure 2.3: Schematic view of the multi-touch panel using frustrated total internal reflection including improvements to increase touch sensitivity.

Additional improvements can be done by adding an infrared blocking filter (which prevents infrared light from the environment reaching the camera) and a protective foil (to protect the projection screen from getting scratched). With these extra layers the input video signal is very clean. Therefore only a few stages of image processing are required before touch detection and tracking can be applied.

Compared to other camera based multi-touch techniques, it requires more effort to construct a FTIR panel. Other disadvantages include the robustness of the touch layer and the cost of the compliant and projection materials.

2.2.2 Diffused Illumination

Rear-side Illumination

The second prototype uses a technique called Rear-side Illumination (RI) which is part of the Diffused Illumination (DI) technique. Based on the technique which is used in the HoloWall and MS Surface, a diffuser is attached to the rear side of an acrylic or a glass like material. Instead of illuminating the display material, multiple infrared light illuminators are placed behind the display pointed straight at the diffuser. A part of the infrared light will be diffused, another part will pass through the material. When the display is touched, the fingertip reflects the infrared light back to the camera. This allows the system to detect touch.

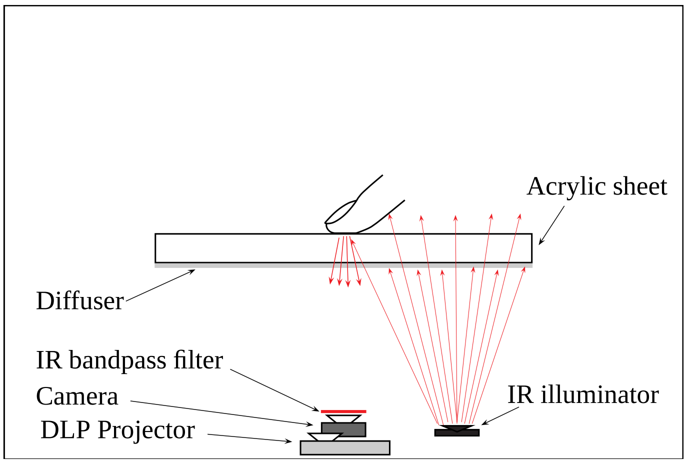


Figure 2.4: Schematic view of the multi-touch panel using rear illumination.

In Figure 2.4 a schematic side view is presented. Compared to FTIR, RI allows us to create a simple and robust multi-touch capable input device. The RI technique provides a few extra features. Besides fingertips it is also capable of detecting objects that reflect infrared light (for example: cellular phones, cards or coffee cups). Another feature is the ability to detect fiducial markers. Depending on the resolution of the used camera and the size of these fiducial markers, it is possible to allow tracking of their position and orientation. This opens up new interaction capabilities in which physical objects can be detected and recognized to trigger specific software functions.

Because the technique is based on reflection rather than changing the refractive index, it works with wet and dry fingers. The performance of RI depends on a few other factors. The most important is the type of diffuser being used (see Appendix A). It is important that the diffuser does not absorb too much infrared light, if it does the contrast between the background and the fingertip will be very low. If the diffuser does not absorb enough infrared light it will be illuminating objects which are near but not on the screen. This would result in false touch detection.

Since the technique relies on contrast differences, the environment in which the multi-touch display is placed has a large influence. Direct sunlight or light sources overhead can decrease the performance of tracking. A captured frame from a RI based devices requires more stages of image processing filters than FTIR before blob detection and tracking can be applied.

Front-side Illumination

Another technique based on diffused illumination is called Front-side Illumination (FI). Like the RI technique it is based on light being diffused. However instead of using infrared light sources, it depends on the ambient light from the environment.

With FI the diffuser is attached to the front side of the display. The ambient light illuminates the diffuser, which from the camera's point of view, results in an evenly colored rectangle. By touching the surface, shadows will appear underneath the fingertips because the ambient light can not reach it.

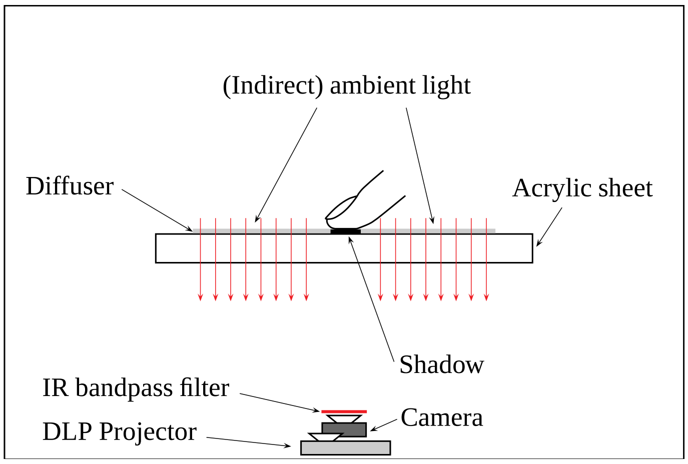


Figure 2.5: Schematic view of the multi-touch panel using front illumination.

Figure 2.5 shows a basic setup for FI. Because the used touch library requires touched spots to be colored white, a simple invert filter is being applied. FI can be seen as the cheapest way of creating a camera based multi-touch capable device. However due to its dependency of evenly distributed light it is less reliable and precise than FTIR and RI.

2.3 Camera devices

A requirement when using FTIR or RI based techniques is that the camera sensor is capable of detecting infrared light. The sensitivity of CMOS/CCD image sensors to infrared light varies. When the camera captures infrared light, the display often shows this as a bright white spot with a blue/dark purple glow. When choosing a camera it is recommended to find out which sensor is used and whether the data sheets are available for this sensor. Most data sheets contain a chapter with the spectral sensitivity characteristics. Often a graph shows how sensitive the sensor is to specific wavelengths. In our case we used illuminators which have a wavelength of 880 nm.

The first prototype used a USB based web camera (Philips SPC900NC) which uses a Sony CCD image sensor (type: ICX098BQ). The spectral sensitivity characteristics are displayed in Figure 2.6.

Consumer cameras such as the web camera of Philips often contain an infrared blocking filter to prevent image distortion from ambient lightning. Since the filter prevents infrared light from reaching the camera sensor, it is required to remove the filter layer. In some cases it is a detachable filter, in other cases it is either glued on to the lens or applied as a coating on the camera sensor. The used Philips camera has the infrared blocking filter glued onto the lens, therefore it was required to replace the original lens with a new lens.

Whilst high-end consumer USB cameras are capable of transmitting images of VGA resolution (640×480 pixels) at reasonable frame rates, they often introduce a latency. Because this reduces the responsiveness of the multi-touch devices it is recommended to use a FireWire based camera instead.

Our second prototype used the *Unibrain Fire-i board camera colour* [35]. The camera uses the same sensor (Sony ICX098BQ) as the Philips camera but due its design it has a lower latency. Depending on the size of the display and the projected image it is recommended to use at least a camera running a VGA resolution because of precision. The frame rate should be at least 30 frames per second to allow smooth interaction.

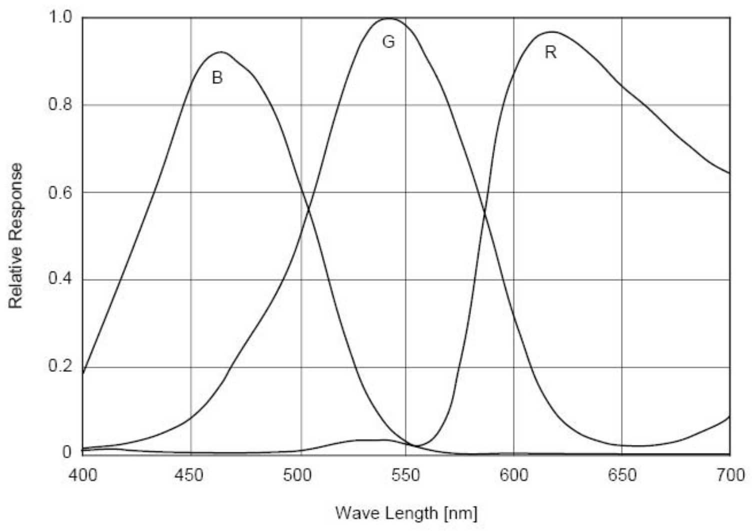


Figure 2.6: The spectral sensitivity characteristics of the Sony ICX098BQ CCD image sensor (excludes lens characteristics and light source characteristics).

Because the camera only needs to see infrared illuminated objects, an IR band pass filter is mounted. When the IR band pass filter is mounted, the projected image is no longer visible. For optimal performance it is recommended to use an (expensive) band pass filter which matches the IR wavelength of the LEDs. Another (cheap) solution is using an overexposed developed negative which performs almost the same as an IR band pass filter.

2.4 DLP projector

When selecting a digital projector it is important to decide which resolution is required. Depending on which type of applications will be running it is recommended to use a resolution of at least 1024×768 pixels (XGA). Depending on the projector type, Digital Light Processing (DLP) or Liquid Crystal Display (LCD), it is important to look at the contrast ratio and the amount of lumen (brightness). Since rear projection is being used the brightness often has to be reduced.

Selecting a digital projector suited for a multi-touch display seemed to be more complex than one would imagine. In most cases (budget) office projectors are not suitable because of their long throwing distance. It is possible to use mirrors to cut down the projection distance, however this reduces the quality and brightness of the image and complicates the design of the device. When mirrors are used it is recommended to use a front surface mirror to prevent a double projection.

We have looked into several (expensive) short throwing distance projectors that are currently available on the market. Based on the specifications and prices (see Table 2.1), the 3M DMS 700 seemed to be the best choice. The 3M DMS 700 is capable of projecting a screen size with a diagonal of 102 cm from a distance of 50 cm.

Projector type	**Native resolution**	**MSRP**
3M DMS 700	1024x768	$ 2765
JVC DLA-SX21SU	1400x1050	$ 11995
NEC WT610	1024x768	$ 3999
Sanyo PLC-XL50	1024x768	$ 2900
Toshiba TDP-ET20U	854x768	$ 999
Toshiba TDP-EW25	1280x800	$ 1899

Table 2.1: Overview short-throw digital projectors. Including manufacturer's suggested retail price (MSRP).

2.5 Choosing a technique

Based on the experiences of building different multi-touch panels, an overview is is presented in Table 2.2

Comparison overview			
Item	**FTIR**	**RI**	**FI**
Component costs	High	Medium	Low
Construction complexity	High	Medium	Low
Closed box required	No	Yes	No
Blob contrast	Strong	Average	Average
Software tracking complexity	Low	Medium	Medium
Reliable finger tracking	High	High	Low
Allows object tracking (pencil)	Yes	No	No
Allows object tracking with fiducials	No	Yes	No
Influence of ambient light	Low	High	High

Table 2.2: A comparison of various aspects in the construction of three different multi-touch panels: Frustrated Total Internal Reflection (FTIR), Rear side Illumination (RI) and Front side Illumination (FI).

Due to the requirements set by us and NEMO, it was decided to build a large table using RI.

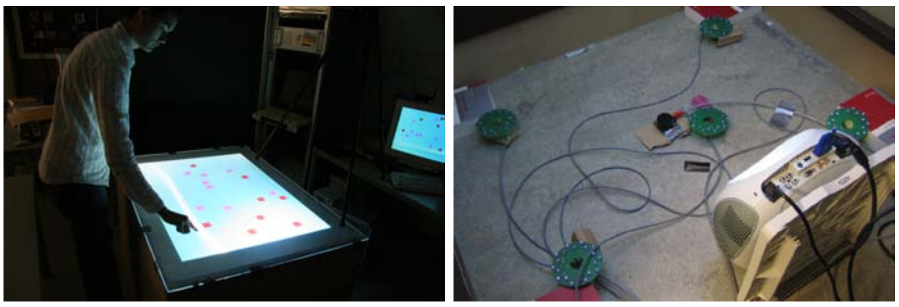
2.6 Hardware description

The host computer consists out of two hyperthreaded Intel Xeon processors running at 3.0 GHz. The computer contains 4.0 GB of memory and uses Windows XP as operating system.

The multi-touch table (see Figure 2.7) uses a FireWire based camera (Unibrain Fire-i Colour) using a F2.5 mm micro lens. The projection on the multi-touch table is created by the 3M DMS 700 digital projector which uses its native resolution (1024$\times$768 pixels). The dimensions of the table are: 120 cm $\times$ 90 cm $\times$ 80 cm (L$\times$W$\times$H) (see Figure 2.8). The actual projection surface is 94.7$\times$72.5 cm (L$\times$W). The diffuser (polyester) is glued onto a sheet of acrylic. In order to illuminate the diffuser, five disc shaped infrared illuminators are used. Each disc contains 20 Osram SFH485P LEDs. The total cost of the system is presented in Table 2.3.

Component	**Price**
Acrylic sheet (120×90×1 cm)	200
Diffuser material	20
Aluminum profile	150
Construction wooden box	500
Digital projector	1500
FireWire Camera with lens and IR bandpass filter	250
IR LEDs	50
Computer	1000
Total hardware costs	3670

Table 2.3: Overview component costs in euros.



(a) Using the multi-touch table (b) Inside the multi-touch table

Figure 2.7: The left image shows the multi-table used by the author of this thesis. The right side shows the main components that are placed inside the multi-touch table. This includes five circular shaped infrared illuminators, a camera with band-pass filter and the digital projector.

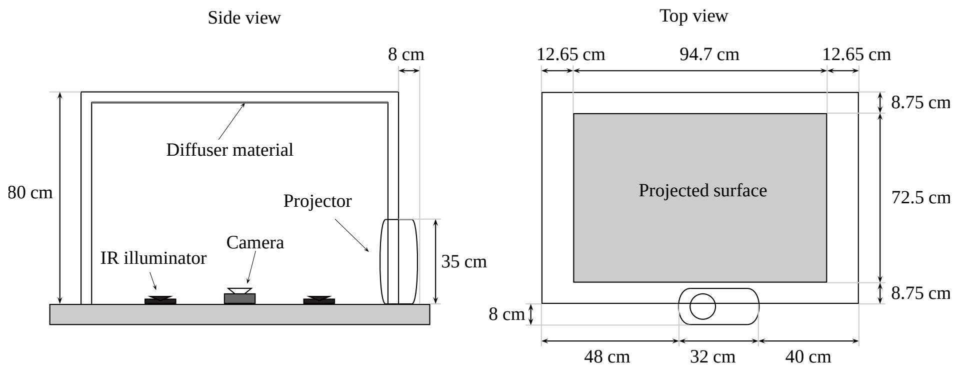


Figure 2.8: Schematic view of the constructed multi-touch table.

CHAPTER 3

Multi-touch detection and processing

To perform camera based multi-touch detection and processing several frameworks are available. In this chapter we describe the used multi-touch framework, how a multi-touch framework connects to a multi-touch application and the different type of gestures.

3.1 Touchlib

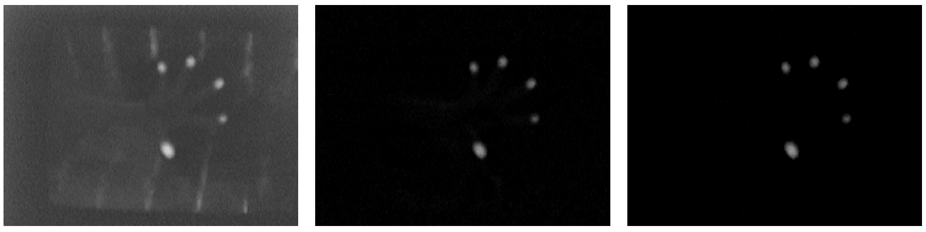
Our multi-touch system uses Touchlib [38] which is a free open source cross platform multi-touch framework which provides video processing and blob tracking for multi-touch devices based on FTIR and DI. Video processing in Touchlib is done through the Intel's OpenCV graphics library [16]. Touchlib currently runs on MS Windows, Linux and Mac OS X.

3.1.1 Video image processing

Frustrated Total Internal Reflection

In the case of FTIR, the video processing chain can be very short. Depending on how much environment 'noise' is present only three filters are required. Depending on the camera, Touchlib provides several options to capture frames. For a cross platform solution it is possible to use OpenCV. Under MS Windows it is recommended to use DSVideoLib [28] (USB and Firewire) or the CMU capture driver [36] (FireWire only) instead. The last two capture filters provide advanced options to set camera modes. Depending the supported camera modes, it is recommended to set it to 8-bit mono, the format that Touchlib filters internally use. An overview of Touchlib filters is available in Appendix B.

When Touchlib is used for the first time (in an application) it stores a copy of the current frame into the memory. This frame is used as a reference frame and used to perform background subtraction on the next frames. The last filter called the rectify filter is used to filter out noise and reduce the gray scale image to a black and white image only displaying the actually touched areas. An example of the Touchlib image processing results can be found in Figure 3.1.

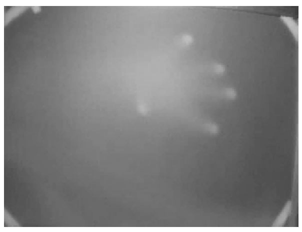


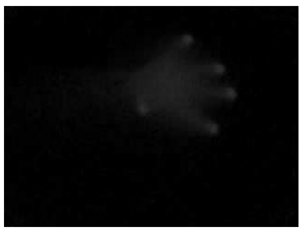
(a) Capture filter (b) Background filter (c) Rectify filter

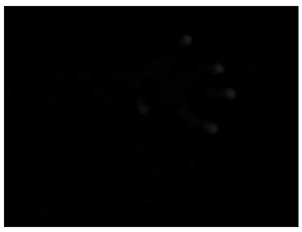
Figure 3.1: Example of the Touchlib filter chain using FTIR.

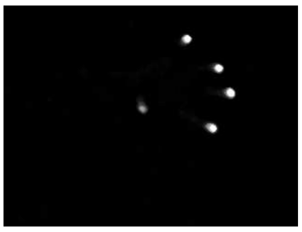
Diffused Illumination

Diffused illumination requires more video processing filters before blob tracking can be applied. First a capture filter is selected depending on the used interface (USB or Firewire). If the system uses FI it is required to add an invert filter because Touchlib expects white colored blobs. The next filter is the background subtraction filter. After removing the (static) background a highpass filter is applied. This filter compares the contrast in the image and only allows 'high' values to pass through (depending on a pre-set value). As a result only the touched spots will be visible. Depending on the used diffuser and infrared LEDs the result might show weak blobs, in this case we can add a scaler filter to amplify the brightness of the blobs. Finally, the rectify filter is applied resulting in clear blobs ready for tracking. An example of the Touchlib image processing results can be found in Figure 3.2.


(a) Capture filter


(b) Background filter


(c) Highpass filter


(d) Scaler filter

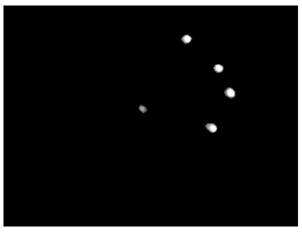

(e) Rectify filter

Figure 3.2: Example of the Touchlib filter chain using RI.

3.1.2 Lens correction

Our current system is using a wide-angle lens which suffers from a radial distortion. Because Touchlib tries to correct a non-linear radial distortion with a linear method based on grid points, it will fail near the corners of the screen. This is the result of the increasing distortion when a point is further away from the image center.

Fortunately multiple algorithms are available to correct a radial distortion caused by the lens. For our system a new Touchlib filter was developed which applies image correction on the video stream. The filter uses standard functions from the OpenCV library [16].

Camera parameters

Before it is possible to correct an image frame it is required to obtain more information about the camera parameters. The camera parameters describe the camera configuration and consists out of two parts, namely the *intrinsic* and *extrinsic* camera parameters.

The intrinsic parameters describe the properties of the camera which include the focal length, the image center (principle point, the effective pixel size and the radial distortion coefficient of the lens. The extrinsic parameters describe the relationship between the camera and the world. These include the rotation matrix and translation vector.

When using a pinhole camera the relationship between a 3D point M and its image projection m is described as follows:

$$m = A[Rt]M. \tag{3.1}$$

In this equation, A represents the camera intrinsic matrix:

$$\mathbf{A} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

c_x, c_y are the coordinates of the principle point and f_x, f_y are the focal lengths on the x and y axis.

The parameters R and t are the extrinsic parameters. R is the rotation matrix and t is a translation vector.

Depending on the quality and size of the used camera lens, lenses often contain a lens distortion which can be characterized by four coefficients (distortion coefficients): k_1, k_2, p_1 and p_2. The first two coefficients describe the radial distortion the last two describe the tangential distortion.

Pattern

In order to find the camera parameters it is required to calibrate the camera by presenting it a known pattern from different angles and positions. In our case we used a chessboard pattern which is obtained from the OpenCV examples. The pattern is printed on an A4-sized paper and glued onto a steady surface.

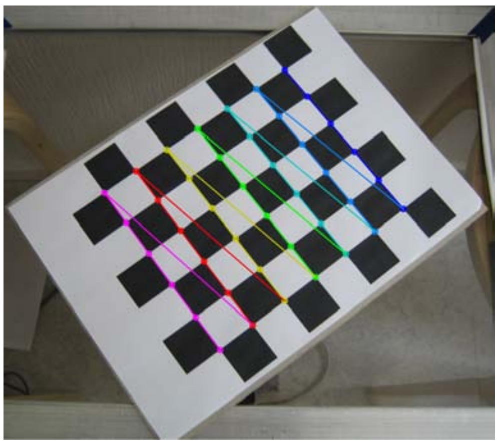


Figure 3.3: Camera image of the chessboard pattern with intersection detected by OpenCV.

By presenting the pattern (Figure 3.3) to the camera to the OpenCV calibration tool, it will recognize the pattern and assign coordinates to the intersections of the chessboard pattern. By showing multiple views of the pattern from different angles, the tool tries to map all the coordinates in a three dimensional space. From the mapping of these points, the tool is capable of calculating the intrinsic parameters and the distortions coefficients.

Correcting lens distortion

Since the required parameters are known, we can now apply the image correction.

According to the OpenCV manual the following is defined:

$$\begin{aligned} x' &= x + x\left[k_1 r^2 + k_2 r^4\right] + \left[2p_1 xy + p_2(r^2 + 2x^2)\right], \\ y' &= y + y\left[k_1 r^2 + k_2 r^4\right] + \left[2p_1 xy + p_2(r^2 + 2y^2)\right], \end{aligned}$$

where:

- $r^2 = x^2 + y^2$,
- x' and y' are the real (distorted) coordinates in the camera image,
- x and y are the ideal (distortion-free) coordinates in the camera image,
- k_1 and k_2 are the radial distortion coefficients,
- p_1 and p_2 are the tangential distortion coefficients.

In this equation the second parts adds the radial distortion and the third part adds the tangential distortion.

However, this equation is not complete. The center of the radial distortion is equal to the principle point. Therefore it is required to adjust the equation as follows:

$$u' = u + (u - c_x)\left[k_1 r^2 + k_2 r^4 + 2p_1 y + p_2\left(\frac{r^2}{x} + 2x\right)\right], \tag{3.3}$$

$$v' = v + (v - c_y)\left[k_1 r^2 + k_2 r^4 + 2p_2 x + p_1\left(\frac{r^2}{y} + 2y\right)\right], \tag{3.4}$$

where:

- u' and v' are real observed (distorted) image coordinates in an ideal projection,
- u and v are true pixel image coordinates in an ideal projection,
- c_x, c_y are the coordinates of the principle point,
- k_1 and k_2 are the radial distortion coefficients,
- p_1 and p_2 are the tangential distortion coefficients.

We use the relation from Equation 3.3 and Equation 3.4 to undistort the image from the camera.

In our custom Touchlib barrel distortion correction filter (Figure 3.4) the function *cvInitUndistortMap* is called with the intrinsic and distortion parameters. This function will create a mapping which is used to translate coordinates from a distorted image to coordinates of the corrected image. These conversion values are stored and only created on initialization. Each captured frame is corrected using the function *cvRemap* with the previous generated conversion mapping. The filter is placed before the last image filter. The resulting image processing pipeline for RI will now look like:

1. Capture filter,
2. Background filter,
3. Highpass filter,
4. Scaler filter,
5. Barrel distortion correction filter,
6. Rectify filter.

3.1.3 Blob detection and tracking

In Touchlib the blob tracker handles the blob detection and tracking. Blob detection is done to detect touch in a camera image. In order to follow the movement of touch, the blob tracker compares the detected touch locations in a frame with the positions of the previous frame.

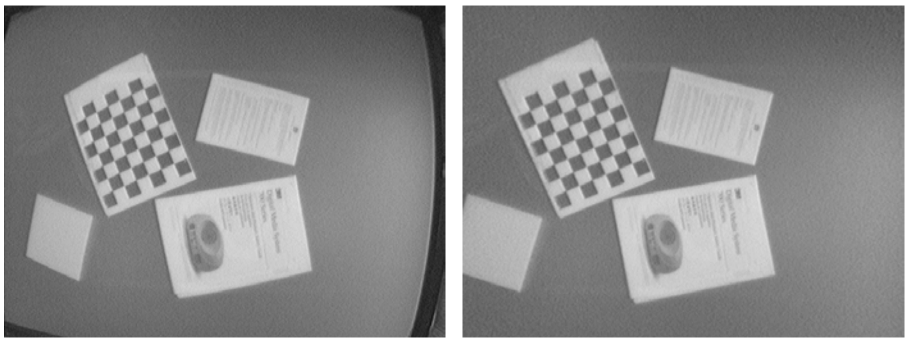

(a) Distorted frame (b) Undistorted frame

Figure 3.4: Example of lens distortion correction. On the left the original camera frame, on the right the corrected frame. As result of the image correction, parts of the original frame are lost.

Blob detection

In each time step, Touchlib requests a frame from the video camera. After a frame has been processed, the resulting image is used to perform blob tracking. By using the OpenCV function *cvFindContours()* we obtain a list of contours found in the current frame. All found contours (and their properties) are stored in a contours list.

Each contour is checked on whether it is a fiducial marker or a touch. On each contour Touchlib tries to fit a polygon which matches the outlines of the contour. If the fitted polygon is build out of four points it might be a possible fiducial marker. Next it will check whether the angle of all the corners matches approximately 90 degrees. If the result is true, the contour is considered a fiducial and the position (center) will be calculated. It will also assign a unique tag identifier based on the pattern found within the square.

If the polygon is more complex than four points, it is assumed to be a touch. Touchlib will fit an ellipse on the contour. The properties of the ellipse are used to determine the position, orientation and the size of a blob. If the size of the blob fits the minimum and maximum requirements on height and width, the blob will be added to the blob list.

Blob tracking

In order to track blobs it is required to have at least two data sets that contain blobs in different states. For our example we first define the two data sets.

The first data set contains the blob list from a previous frame and is defined as follows:

$$p_1, p_2, p_3, \ldots, p_n$$

where n is the number of active blobs in the previous frame.

The second set contains the blobs list of the current frame and is defined as follows:

$$c_1, c_2, c_3, \ldots, c_m$$

where m is the number of active blobs in the current frame.

After each frame, the data from set p is replaced by set c. Set c will be filled with the new blob list.

Example:
We define an example with two states represented in Figure 3.5.
The data set of the previous frame:

$$p_1 : (1, 4), p_2 : (3, 1) \Rightarrow n = 2.$$

The data set of the current frame:

$$c_1 : (3, 4), c_2 : (2, 3), c_3 : (1, 1) \Rightarrow m = 3.$$

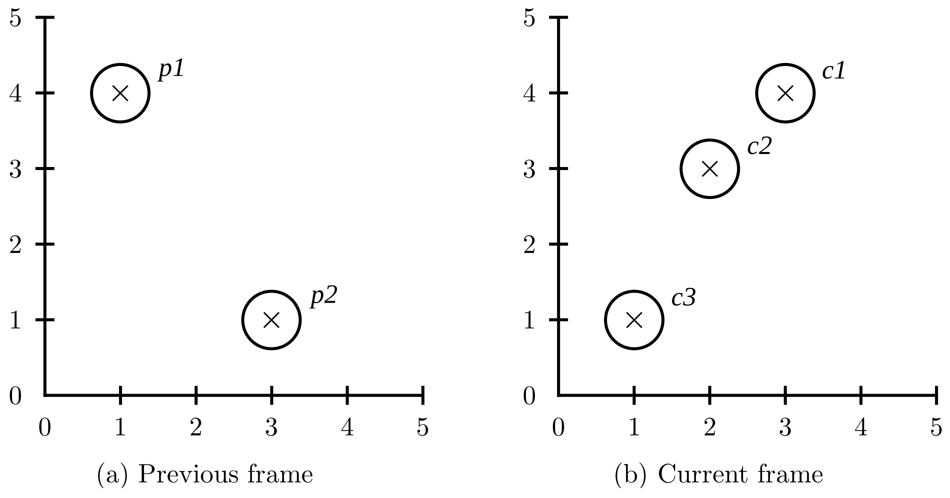


Figure 3.5: An example of two frames containing different blob states.

In order to match blobs in the previous and current states it is required to create a matrix that contains the possible conversion states. The number of possible states can be calculated using the following equation:

$$N = \frac{(n+x)!}{(n+x-m)!x!}, \tag{3.5}$$

where $x = m - n$.

In our example $x = m - n = 1$. According to Equation 3.5 the number of possible states is:

$$N = \frac{(2+1)!}{(2+1-3)!1!} = 3! = 6.$$

For each possible state an entry is added into the transition matrix (Table 3.1).

state	c_1	c_2	c_3
s_1	new	p_1	p_2
s_2	new	p_2	p_1
s_3	p_1	p_2	new
s_4	p_1	new	p_2
s_5	p_2	p_1	new
s_6	p_2	new	p_1

Table 3.1: The transition matrix showing all possible transition states.

Touchlib's blob tracker tries to find a state which contains the lowest distance value. This value is calculated by comparing the distances between the blobs out of set p and set c. In our example set c contains more values than set p, which means that the current frame contains a new blob. When calculating the distance value, the new blob will be assigned the value of zero. Based on the transition matrix we can calculate the following distance values:

$$\begin{aligned}
s_1 &= 0 + \sqrt{(2-1)^2 + (3-4)^2} + \sqrt{(1-3)^2 + (1-1)^2} = \sqrt{2} + \sqrt{4} \approx 3.41,\\
s_2 &= 0 + \sqrt{(2-3)^2 + (3-1)^2} + \sqrt{(1-1)^2 + (1-4)^2} = \sqrt{5} + \sqrt{9} \approx 5.24,\\
s_3 &= \sqrt{(3-1)^2 + (4-4)^2} + \sqrt{(2-3)^2 + (3-1)^2} + 0 = \sqrt{4} + \sqrt{5} \approx 4.24,\\
s_4 &= \sqrt{(3-1)^2 + (4-4)^2} + 0 + \sqrt{(1-3)^2 + (1-1)^2} = \sqrt{4} + \sqrt{4} \approx 4.00,\\
s_5 &= \sqrt{(3-3)^2 + (4-1)^2} + \sqrt{(2-1)^2 + (3-4)^2} + 0 = \sqrt{9} + \sqrt{2} \approx 4.41,\\
s_6 &= \sqrt{(3-3)^2 + (4-1)^2} + 0 + \sqrt{(1-1)^2 + (1-4)^2} = \sqrt{9} + \sqrt{9} \approx 6.00.
\end{aligned}$$

From these results Touchlib will choose the row with the lowest distance value. In our example this is s_1 which corresponds to Figure 3.6. It is possible that multiple rows share the same distance value, in this case Touchlib chooses the first row containing this value (if this value is considered the lowest distance).

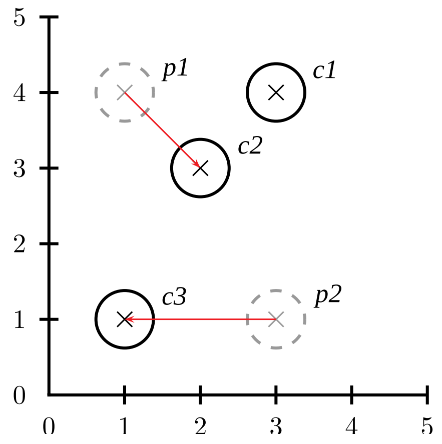


Figure 3.6: Final state blob tracker.

Position correction with barycentric coordinates

When using a camera with a wide angle lens, the camera image suffers from radial image distortion (barrel distortion). The amount of distortion depends on the size and focal length of the lens. Fortunately Touchlib provides a basic coordinate correction system which is capable of correcting light radial distortions. In order to do so, Touchlib uses the barycentric coordinates system [39, 40].

We consider the example in Figure 3.7. In this figure point P represent a touch in the triangle ABC. The barycentric coordinates are named as α, γ and β. These values are normalized so they become the areas of the subtriangles which connect to P (PAB, PBC and PCA). The relation between the coordinates can be described as:

$$\alpha + \gamma + \beta = 1.$$

This relation is used to correct the (distorted) coordinate of point P in Figure 3.7.a to the (corrected) coordinate in Figure 3.7.b.

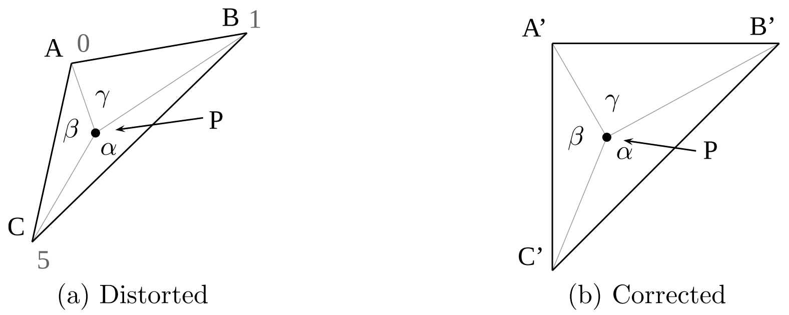


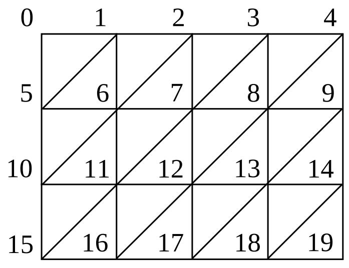
Figure 3.7: Correcting image distortion with barycentric coordinates mapping.

Touchlib tries to map a triangle grid on the camera image. An example situation is presented in Figure 3.8.

During the calibration step the user is required to touch the points in the grid to set the position. Because the distortion is radial, the grid points will not be aligned evenly. Instead it follows a radial curve which is visible in Figure 3.9.

Generating events

In the last part of the blob tracking, Touchlib prepares the events to be triggered.



(a) Calibration grid

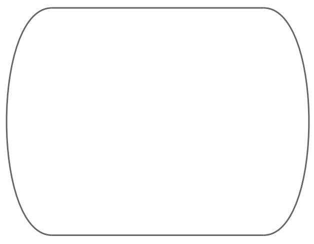

(b) Camera frame

Figure 3.8: An example of the calibration grid and a camera frame suffering from barrel distortion.

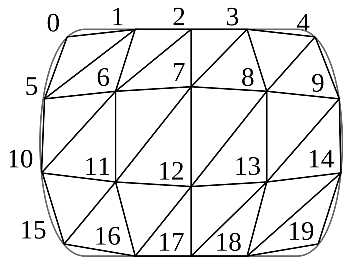


Figure 3.9: Calibration grid mapped on camera frame.

First Touchlib walks through the list of currently active blobs. If a blob is new in the current frame (it did not match with a blob in the previous frame) it will contain an ID with the value -1. In that case a new ID will be generated and assigned. After the assignment a “touch event” of the blob will be dispatched.

If a blob in the current frame matches a blob of the previous frame, it will use the same identifier of the previous frame. Depending on the amount of movement an update event will be dispatched. If the delta of movement is higher than the set value of *distanceThreshold* it means the blob ‘traveled’ over a larger distance than one would assume to be correct. In this case instead of an update event a “touch event” is dispatched instead. If the delta of movement from the current and the previous frame is below the value of *minimumDisplacementThreshold*, no update event will occur.

After Touchlib processed the list of currently active blobs, it compares the list with the blob list of the previous frame. If a blob was active in the previous frame but has not been found in the current frame, the blob is marked for deletion and a “touch up event” will be dispatched.

3.2 Programming language interfaces

In order to use Touchlib with other languages than C++, several wrappers are available which allow applications to receive data from Touchlib.

3.2.1 TUIO/OSC protocol

By default Touchlib comes with a wrapper which sends TUIO events over the commonly used OpenSound Control (OSC[1]) protocol. For many modern programming languages such as C#, Adobe Flash (Actionscript 3), Java, Max/DSP, Processing, Pure Data, Python and Visual Basic, OSC libraries are available. When using Flash it is required to convert UDP packages to TCP. This can be done by using a tool called Flosc which acts as a proxy (Figure 3.10).

When an application uses the OSC protocol, it will only be able to receive events containing properties of the detected blobs. It is not possible to adjust the settings of Touchlib from the application. Since OSC uses the UDP network protocol to transfer

[1] http://www.cnmat.berkeley.edu/OpenSoundControl/

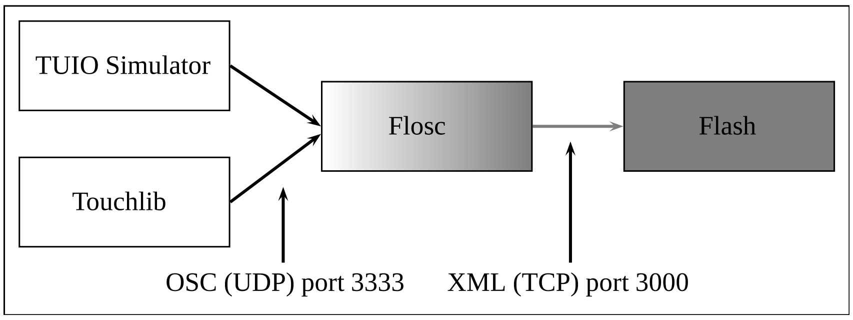


Figure 3.10: Schematic view of sending TUIO events over the OSC protocol. Flosc converts the UDP packages to TCP packages.

data it makes it possible to create a setup in which a dedicated system provides blob tracking and transfers the data to another system which provides the visualization.

3.2.2 C# wrapper

When using a C# based multi-touch application is being designed it is recommended to use the TouchlibCOMwrapper. The TouchlibCOMwrapper provides a clean and easy to use framework to use Touchlib in C#. Since the TouchlibCOMwrapper communicates with Touchlib directly, it is possible to use most functions of Touchlib such as requesting a status update and recapturing the background (in case the ambient light situation changed). The wrapper uses the Microsoft Component Object Model (COM[2]) API to communicate with Touchlib, therefore it is only usable with MS Windows based operating systems.

3.3 Gesture based interaction

In comparisons with traditional desktop computing with mouse and keyboard, a multi-touch device provides additional input methods. Instead of working as a point and click device it can be improved with gesture based interaction.

3.3.1 Gesture classification

In order to distinguish different gesture the following gesture are defined: direct gestures and symbolic gestures. We use the same naming convention as used by Oka et al. [27].

Direct gestures

In the definition of *direct gestures* we describe gesture patterns that allows a user to manipulate objects directly. The manipulation can be performed with multiple pointers. For example, when using a multi-touch system it is very common to use a pinching like motion in order to change the size of an object. Depending on the distance between two fingers, the object increases or decreases in size. Examples of common used direct manipulation gestures are shown in Figure 3.11.

Symbolic gestures

Symbolic gestures are patterns based on the gesture location and trajectory. Patterns can be made in any shape such as triangles, circles or even text characters. Symbolic gestures are commonly used in applications to simplify tasks. An example can be found in modern web browsers that feature mouse gesture recognition. When pressing a pre-defined button, the mouse gesture recognition mode is enabled. The user can now draw a gesture on the browser window. After releasing the button, the gesture

[2]http://www.microsoft.com/com/default.mspx

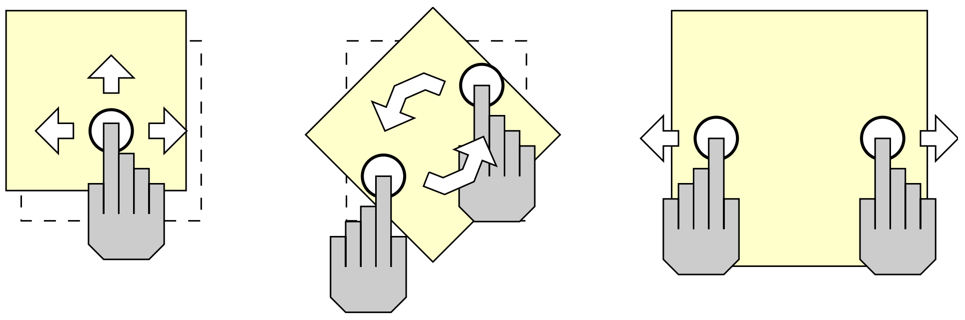


Figure 3.11: A common used direct manipulation gesture set used for object manipulation.

recognition engine will try to match the gesture pattern and complete the task which the gesture is bind to (such as switching between pages and tabs).

3.3.2 Gesture recognition models

In order to perform gesture recognition, different recognition models are available. We discuss the most common used models. Based on the advantages and disadvantages we select a technique that will be used for Gesturelib.

Neural networks

Neural networks are commonly used for pattern recognition [3]. A basic feed forward neural network (FFNW) consists out of three layers. The input layer, the hidden layer and the output layer.

Before it is possible to use a FFNW for gesture recognition, it is required to define a data format which will be fed to the FFNW. For symbolic gestures path information is stored as a list of x and y values. Depending on the precision of the system, the path list can increase in size rapidly. To reduce the complexity, the number of entries is reduced (for example 16 entries in total). The entries in the path list will now represent a rough estimation of the original path. For each point in the path the angle is calculated.

The input layer of a FFNW contains 16 neurons, the same number as points we reduced our path list to. Each angle value is passed to one neuron of the input layer. Internally each neuron from the input layer is connected to all of the neurons in the hidden layer. The number of neurons in this layer does not strictly have to be 16 as well. The neurons of the hidden layer contain a value (weight). Next each of the neurons in the hidden layer is connected to one of the neurons of the output layer. The number neurons in the output layer depend on the number of gestures that are required to be recognized.

When a gesture is performed a list of angles is passed to the FFNW. The value of the input layer is now multiplied with the value of the hidden layer. The result is passed to the output layer. Since each gesture results in a unique value in the output layer, it is possible to recognize patterns.

The downside of using FFNW is that the system needs to be trained. On initialization the weights in the hidden layer are filled with randomly chosen values. When training the FFNW with gesture pattern the weight values in the hidden layer are adjusted. The training is completed when each pattern only triggers one neuron in the output layer.

Region based

A popular gesture recognition program for the X Window System is Xstroke [46]. Xstroke is a full-screen gesture recognition program which allows users to bind commands to a gesture.

The technique which Xstroke uses to recognize gestures is region based. Internally Xstroke holds a 3×3 grid which is numbered from 1 to 9. The grid is centered on the gesture and will scale to the extends of the obtained gesture path. For example, if the user draws an N shaped gesture, the line will pass the following regions: 7, 4, 1, 5, 9, 6, 3. Xstroke stores the values of the passed grid cells in a string. By comparing the result by regular expressions with the gestures in the database, gestures can be detected.

Direction based

The last gesture recognition technique that will be discussed is based on direction [2]. When drawing a gesture, the path can be described as a sequence of directions. In our example we limit the number of possible direction to eight. Each direction is numbered from 0 up to 7 (counted in clockwise order, the first direction is pointing to the right and equal to zero). When drawing a gesture shaped as an N, the ideal direction sequence would be “616” which is equal to up, downright, up. Depending on the accuracy of the input device and user, the recorded sequence might match or not. In order to compare both sequences the Levenshtein [21] cost method is used. The Levenshtein cost method compares the recorded sequence with the list of known sequences from the gesture library. For each entry in the gesture library a Levenshtein distance value will be calculated which is based on how similar both strings are. The match with the lowest Levenshtein cost will be selected as the preferred match. By specifying a maximum cost value it is possible to discard wrong matches.

3.3.3 Gesturelib

Gesturelib is a gesture recognition library that uses the direction based recognition technique. We chose the direction based method because it is easy to implement and the technique does not require training. To add Gesturelib to a Touchlib based application it is required to add additional hooks to the *ITouchListener* class. These hooks pass touch data to Gesturelib. An example can be found in Listing 3.1.

```cpp
class ExampleApp : public ITouchListener
{
    public:
        ExampleApp()
        {
            glib.Init("mtgesture.xml");
        }

        ~ExampleApp();

        virtual void fingerDown(TouchData data) {
            touch_list[data.ID] = data; // Store touch data
            glib.TouchDown(&data.ID, &data.X, &data.Y);
        }

        virtual void fingerUpdate(TouchData data) {
            touch_list[data.ID] = data; // Update touch data
            glib.TouchUpdate(&data.ID, &data.X, &data.Y);
        }

        virtual void fingerUp(TouchData data) {
            touch_list.erase(data.ID); // Remove touch data
            glib.TouchUp(&data.ID, &data.X, &data.Y);
        }

        std::map<int, TouchData> touch_list;
        gesturelib glib;
}
```

Listing 3.1: Adding Gesturelib hooks to the ITouchListener object.

On initialization Gesturelib reads the file mtgesture.xml. This file contains settings such as screen dimensions and tolerance settings for Gesturelib. By default it requires a minimum gesture size (path) of sixteen points. The file also contains a list of known gesture patterns. Each gesture entry contains an identifier for the recognized gesture and the string based on the path direction.

When a gesture is being created, Gesturelib will keep recording the path until *glib.TouchUp* is called. When this function is called, Gesturelib will start processing the path. If enough path data is available, it will start calculating the path directions and compare the result with the gesture pattern database. The result is returned to the main application. The application now decides how to process this result.

An overview of the event data pipeline is shown in Figure 3.12.

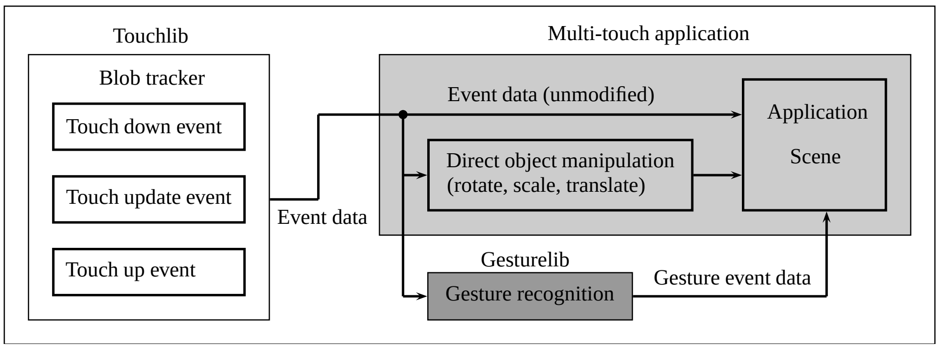


Figure 3.12: Schematic view of the events pipeline using Touchlib and Gesturelib in an application.

CHAPTER 4

Multi-touch applications

Multi-touch technology allows users to interact with a computer in a new way. When using a multi-touch device, the type of interaction methods depend on the application. Current operating systems do not support multi-touch natively. Therefore it is required to implement multi-touch callbacks in the application code manually. This chapter describes how multi-touch applications are created and how existing (single touch) applications can be improved with multi-touch.

4.1 Software architecture

Depending on the used software language, different methods are used to implement a multi-touch capable application. Figure 4.1 presents the software architecture of C++, C# and Flash based applications. In order to use multi-touch in a C# application, the touchlibCOMwrapper is used which provides a two way interface with Touchlib. When using Flash, it is only possible to receive blob positions from Flosc. It is not possible to control Touchlib from the Flash application or Flosc.

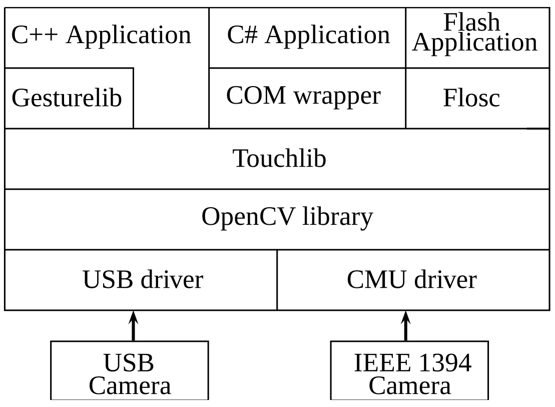


Figure 4.1: A block diagram presenting the software architecture of C++, C# and Flash based applications.

4.2 Multi-touch example

To demonstrate the structure of a multi-touch application we start out with a basic C++ framework of a graphical single touch application (Listing 4.1).

```
int main()
{
    // ... Setup scene and initial object positions ...
    do {
        get_keyboard_mouse_input();
        set_object_positions();
        display_scene();
    } while( running );
}
```

Listing 4.1: An example of a single touch application.

This example contains a basic application framework with a program loop. The program loop includes a function which requests the state of the input devices, a function which sets the object positions and a function which displays the objects on the screen.

Since we want to be able to manipulate the object with multi-touch we add multi-touch support by using the Touchlib framework. We add two objects to the example applications namely, *ITouchScreen* and *ITouchListener*. The *ITouchScreen* object handles the image processing and blob tracking. The *ITouchListener* object handles the events from *ITouchScreen*. This object contains three mandatory functions (*fingerDown*, *fingerUpdate* and *fingerUp*), an example is given in Listing 4.2. If it is required to access the touch information from functions that are not in this class, we use a variable such as *touch_list* that keeps track of the currently active blobs. More information about the TouchData structure can be found in Appendix B.5.

```
class ExampleApp : public ITouchListener
{
    public:
        ExampleApp();
        ~ExampleApp();

        virtual void fingerDown(TouchData data) {
            touch_list[data.ID] = data; // Store touch data
        }

        virtual void fingerUpdate(TouchData data) {
            touch_list[data.ID] = data; // Update touch data
        }

        virtual void fingerUp(TouchData data) {
            touch_list.erase(data.ID); // Remove touch data
        }

        std::map<int, TouchData> touch_list;
}
```

Listing 4.2: Implementing the ITouchListener object.

Next the *ITouchScreen* needs to be initialized. If available it will read the camera settings and the filter pipeline from a configuration file. If the configuration file is not available it will use a default setting. In order to connect the *ITouchScreen* with the application we register the *ITouchListener* to the *ITouchScreen* object. Each time an update is requested by the application (with *getEvents*) the *ITouchScreen* object will call *ITouchListener* with event updates. After completing initialization we can start the image processing pipeline and the blob tracker.

In the program loop we add *getEvents* which request updates from Touchlib. Depending on the required interaction methods object positions can be modified according to the position of blobs stored in the *touch_list*. If it is desired to manipulate objects by direct manipulation, this can be done by checking the number of blobs that are active on an object. By storing the blob data and identity we can compare the position of the current with the previous state. Depending on the change of position and distance the object can be manipulated.

The final result of the example application is in Listing 4.3.

```
int main()
{
    // ... Setup scene and initial object positions ...
    ITouchScreen *screen; // ITouchScreen object
    ExampleApp myapp;     // ITouchListener object

    // Setup Touchlib
    load_configfile();
    register_itouchlistener();
    start_image_processing();
    start_blob_tracker();

    do {
        get_keyboard_mouse_input();
        screen->getEvents();
        set_object_positions();
        display_scene();
    } while( running );
}
```

Listing 4.3: The example application including additions to process multi-touch events.

4.3 Multi-touch demo applications

4.3.1 tDesk

Inspired by the Multi-Point X Server [15] (which is a modification of the X Server that allows users to use multiple mice and keyboards simultaneously under Linux) we created tDesk. Currently MS Windows operating systems do not support multi-touch natively. With tDesk we demonstrate how multi-touch can be used on a MS Windows based operating system. Due to its design it only allows users to organize and resize multiple windows by using direct gestures. When tDesk is running in multi-touch mode, it does not allow the user to use the actual application. In order to use the application a second mode, mouse mode, is added. The mouse mode is a mouse simulator which is capable of simulating a mouse with one mouse button (left). This allows the user to use the multi-touch table as a touch screen.

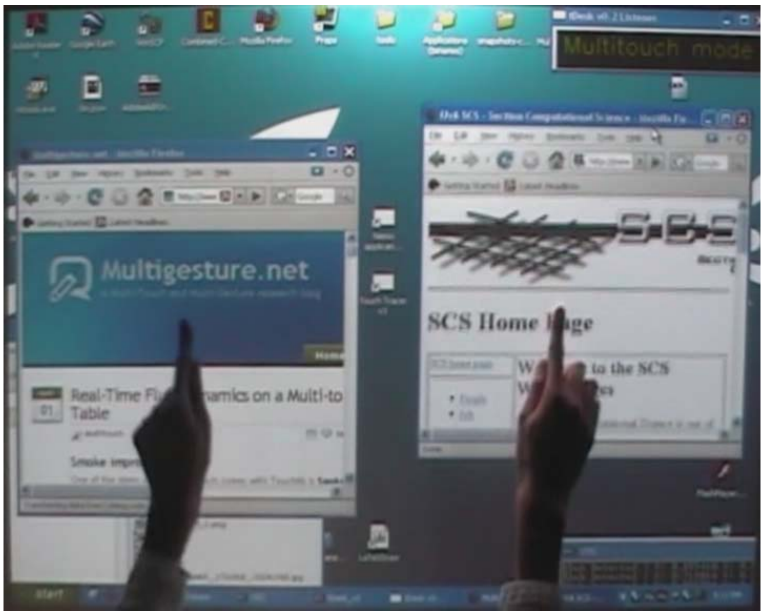


Figure 4.2: tDesk running in multi-touch input mode.

4.3.2 Real-time Fluid Dynamics Simulation and Manipulation

One of the benefits of multi-touch panels is that it allows direct interaction with the scene. Based on the fluid solver [32, 33] by Jos Stam a custom version was created that allows multi-touch input interaction.

By default the application loads a pattern from an image file. The pattern is a black and white image in portable bitmap (PBM) format. White pixels are considered a boundary. Three static sources are placed in the scene by default. The application contains two interaction modes. In the first mode (simulation mode), touch is considered as a source and will insert a randomly colored fluid into the simulation. If the velocity of the source created by touch is high enough, it is possible to frustrate the static sources.

In the second mode (sandbox mode) it is possible to change the scene by drawing new boundaries. By drawing boundaries near the static sources, it is possible to see how fluid flow acts in real-time on these boundaries.

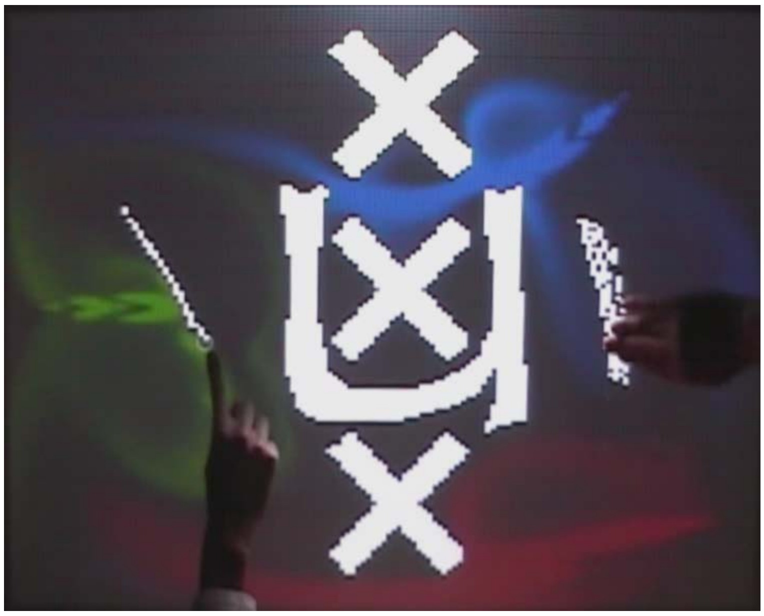


Figure 4.3: Real-time Fluid Dynamics Simulation running in sandbox mode. Two users are drawing boundaries while the simulation is still active.

4.3.3 NASA World Wind

NASA World Wind (NWW) [6] is a geographical application that allows a user to explore the world from top down view. Based on the idea that has been demonstrated by Jeff Han, a custom version of NWW was developed. Unlike Google Earth, NWW is an open source project which allowed us to add multi-touch interaction technology into the application. When using NWW as a normal desktop application, controlling the view depends on mouse interaction (and optional keyboard modifiers). By using multi-touch we changed the way of interacting with NWW. Instead of performing mouse simulation we define gestured based interaction using direct manipulation. We used the TouchlibCOMwrapper to interface with Touchlib.

To keep it simple, two gestures were defined. The first gesture (single touch) allows the user to move the globe by touching the surface and moving the finger. The second gesture (two points) allows the user to control the zoom level (by pinching) and rotation.

Compared to our previous application, this application shows that multi-touch interaction does not mean that the application becomes multi-user as well. Due to the limitation of controlling one camera viewport NWW stays a single user application, even when using multi-touch technology.

4.3.4 Multi-touch media application

The Multi-touch Media Application (MMA) demonstrates the use of gesture based interaction in a multi-user environment. Originally designed as a multi-touch test application, it allows users to watch, sort and share video and photo content. It is possible to load photos while the application is running by inserting a memory card

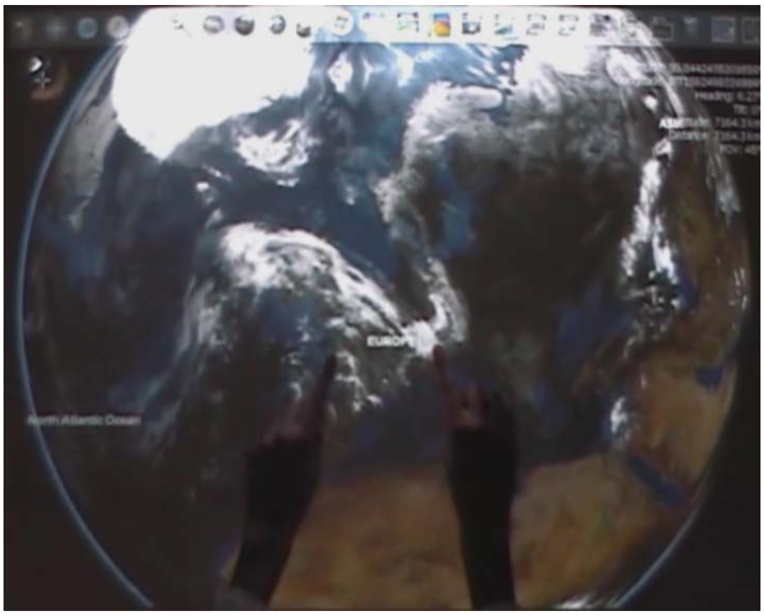


Figure 4.4: Controlling the viewport of NASA World Wind with multi-touch.

in the computer. The application also features a virtual keyboard with a note block and a Google maps widget.

MMA is created with actionscript 3.0 (Adobe Flash) and runs on any system supporting the Adobe Integrated Runtime (AIR, a cross-operating system runtime environment). When Flash is used, it is required to use the TUIO classes provided in the Touchlib framework (Figure 4.1). These classes are used to receive and translate the TUIO messages which are sent by OSC.

A benefit of using Flash as a platform to construct multi-touch applications, is that it is easy to use. Flash CS3 comes with a large set of classes that handle audio and video manipulations. Listing 4.4 shows the basic structure of a Flash based application.

```
 1 import whitenoise.*;     // TUIO classes
 2 import flash.display.*;
 3 
 4 public class MMA extends Sprite
 5 {
 6     public function MMA()
 7     {
 8         TUIO.init(this, "127.0.0.1", 3000, 1024, 768);
 9 
10         var MainCanvas = new PhotoCanvas();
11         this.addChild(MainCanvas);
12 
13         var Photo1:Photo = new Photo("image1.jpg");
14         MainCanvas.addChild(Photo1);
15 
16         var myKeyboard:OnScreenKeyboard = new OnScreenKeyboard();
17         MainCanvas.addChild(myKeyboard);
18     }
19 }
```

Listing 4.4: Example multi-touch application using actionscript 3.

In the first line we import the TUIO classes, the second line imports the classes required for Display and Sprite objects. A basic application structure is created as a class. On line eight we initialize the TUIO class, a reference to *this* application class is passed, the ip address of the Flosc service, the port of the Flosc service, the display width and the display height. Next a main canvas is added to the scene. The canvas is an object based on the RotatableScalable class. The RotatableScalable class is part of Touchlib's actionscript library and implements direct gestures as described in Section 3.3.1. In this example we also added a photo object and an on screen keyboard object (both based on the RotatableScalable class) on the canvas. The application code is now completed.

During testing of the virtual keyboard we experienced the low input performance. While virtual keyboard object can be scaled and locked according to the user's preferences, typing text on the virtual keyboard did not work out well. Unlike normal keyboards the virtual keyboard does not provide tactile feedback. Since most computer users do not look at the keyboard for each word they type, it is hard to find out whether a key has been pressed and whether this was the right key. It is possible to add audio feedback for the key press.

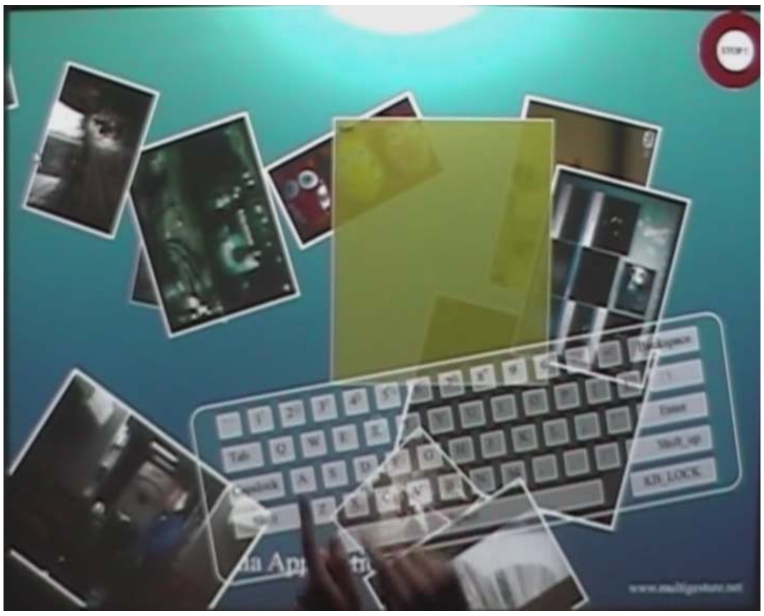


Figure 4.5: Example scene of the Multi-touch Media Application. On the canvas photo and video objects are present. The virtual keyboard is used to type text in the yellow box.

4.3.5 Video puzzle application

In order to demonstrate how multi-touch can be used in collaborative problem solving we developed a digital version of the classic jigsaw puzzle. Like normal jigsaw puzzles the game allows multiple people to solve the puzzle collaboratively. What makes this puzzle game different is the fact that we use a video instead of a picture. During the game the video keeps playing on each of the tiles. This increases the difficulty of the game but also makes the game more amusing.

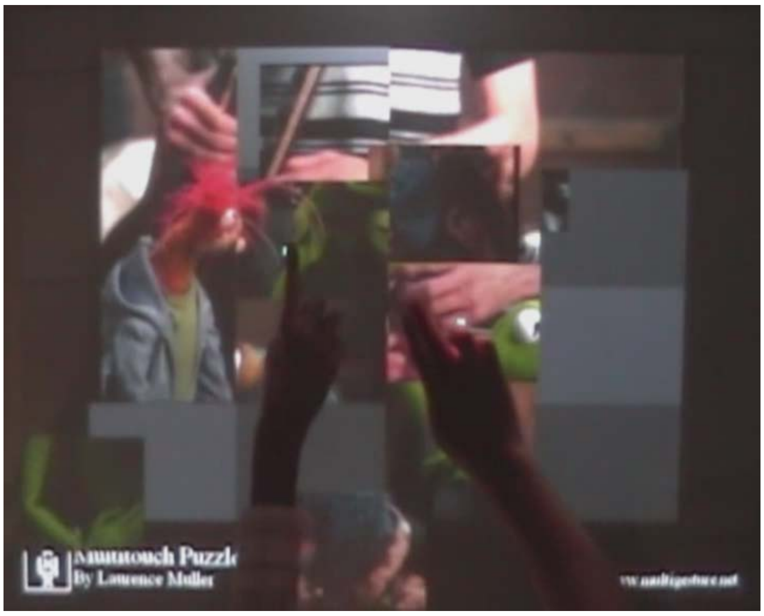


Figure 4.6: Two users trying to solve a puzzle in the video puzzle application.

CHAPTER 5

Multi-touch task performance measurements

With the introduction of multi-touch display devices a new way of human-computer interaction has been introduced. This chapter focuses on comparing input devices on different types of tasks and on whether collaborative work on a multi-touch can improve the performance of a task. One of the most important questions that arise with new input devices is whether they are capable of simplifying common tasks that users perform on computers, and in which level they are capable of replacing existing input device technologies.

Before designing the experiments we formulate the following hypotheses:

1. The performance of a task on a multi-touch device depends on the performance of the used hardware.

2. Some tasks can be performed faster on a multi-touch device than a mouse.

3. Collaboration on a multi-touch device allows users to complete a task faster than completing a task alone.

In order to investigate these hypotheses experiments were created. To test the hardware, multiple benchmark tools were designed. For the performance measurements four different experiments have been designed. The first two experiments focus on comparing the performance of a mouse device with a multi-touch display device. The last two experiments compare the impact of collaboration on task performance.

5.1 System latency

In order to measure the influence of hardware on the performance of a multi-touch device, we measured each part of the system illustrated in Figure 2.1 separately.

5.1.1 Camera

In the first stage of the systems pipeline a camera frame is acquired for processing. It is possible to calculate the delay according to the Instrumentation & Industrial Digital Camera (IIDC[1]) specifications. Each frame is transferred through the FireWire bus. The time it takes to transfer a frame depends on the format at which the camera operates. The FireWire bus sends out packages in 125 microsecond time intervals. The camera operates at 8 bit monochrome with a resolution of 640×480 pixels at 30 frames per second. According to the IIDC specifications this format uses 240 packets per frame (1280 bytes per packet). The time needed for each frame to be transmitted to the system is 240×125 μsec = 30 milliseconds.

5.1.2 Touchlib

In the next stage of the pipeline the camera image is processed by Touchlib.

[1] http://damien.douxchamps.net/ieee1394/libdc1394/iidc

Latency measurement method

In order to measure the latency of Touchlib, timers were added to the Touchlib source code. Each image filter was measured separately. All tests were done on the same machine used for the experiments.

Latency results

During the implementation of the timers in the source code a ‘bug’ in Touchlib was discovered which influence the performance of Touchlib. After each image processing and blob tracking loop, the loop stalls for 32 ms (*Sleep(32)*). This means that the actually processing time of Touchlib is the total Touchlib time from Table 5.1 plus 32 ms resulting in an average of 65 ms.

Filter type	**No active blobs**		**Five active blobs**	
CMU capture	3.351 ms	10.49%	3.048 ms	9.63%
Background removal	0.569 ms	1.78%	0.565 ms	1.78%
Simple highpass	4.751 ms	14.87%	4.788 ms	15.12%
Scaler	2.767 ms	8.66%	2.806 ms	8.86%
Barrel distortion correction	19.962 ms	62.49%	19.913 ms	62.90%
Rectify	0.544 ms	1.70%	0.538 ms	1.70%
Total filter time	**31.944 ms**	**100%**	**31.658 ms**	**100%**
Finding blobs	1.276 ms	99.61%	1.491 ms	94.97%
Tracking blobs	0.004 ms	0.31%	0.061 ms	3.89%
Dispatching events	0.001 ms	0.08%	0.018 ms	1.15%
Total tracker time	**1.280 ms**	**100%**	**1.570 ms**	**100%**
Total Touchlib time	**33.225 ms**		**33.228 ms**	

Table 5.1: Touchlib image processing and blob tracker latency results.

5.1.3 Digital projector

In the last stage of the pipeline the resulting image is presented to the user by the digital projector.

Latency measurement method

Based on the latency measurement tool described in the paper [30] a re-implementation was created.

The measurement tool allows us to compare the number of frames the digital projector lags compared to the reference CRT monitor. In order to use the tool it is required to turn on vertical synchronization in the display settings. For our measurements we used an Iiyama Vision Master Pro 510 as reference CRT monitor. When starting the tool it is required to input the native resolution and refresh rate of the digital projector. The 3M DMS 700 uses a resolution of 1024×768 at 60 Hz. While the tool is running a large green bar will move position each time a frame is being refreshed. In our case that means 60 positions per second.

To perform the measurement the display of the CRT is cloned to the native resolution and refresh rate of the projector. Next the projector is aimed at a projection surface place next to the CRT monitor. To compare the latency it is necessary to take a photograph that includes the image of the CRT and digital projector in one frame. In order to take this picture, a digital camera was used with the shutter speed set to the refresh rate of the screen. The latency in milliseconds can be calculated by comparing the position of the green bar of both devices (see Figure 5.1).

Latency results

We performed measurements on multiple digital projectors. The resulting images were compared on the computer. The image from the 3M DMS 700 (Figure 5.1)

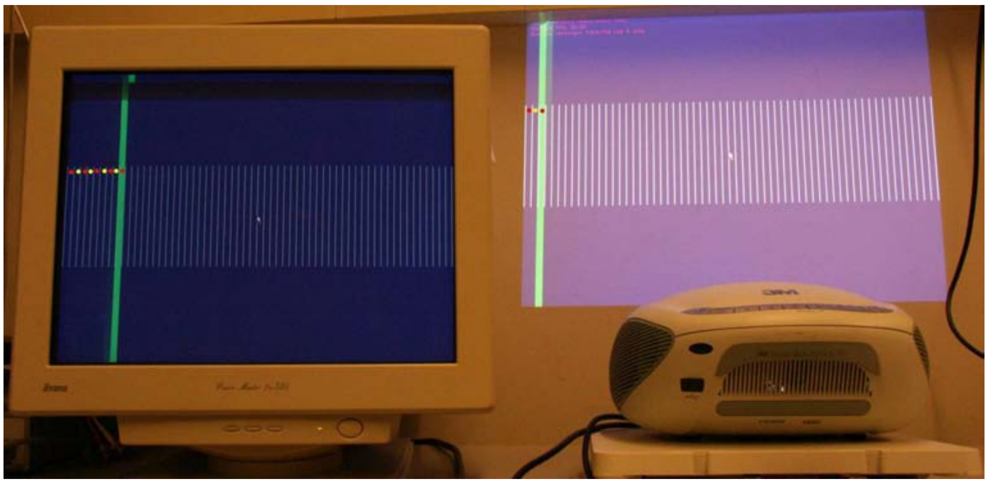


Figure 5.1: Comparing the latency of the digital projector with a CRT monitor using the latency tool.

showed a delay of six frames. Since the refresh rate was fixed at 60 Hz, the refresh time of a single frame is equal to $\frac{1}{60}$ seconds $\approx$ 16.667 ms. By multiplying this value with the number of (delayed) frames we can find the following latency:

$$\frac{1}{60} \times 6 = 0.1\ seconds = 100\ milliseconds.$$

After processing the results of the other digital projectors we found out that only the 3M DMS 700 had such a high latency. After contacting 3M support, they informed us this was probably due the image improvement chip (Hollywood Quality Video, HQV). According to 3M it was not possible to turn off this image processing chip. Results of projectors tested by us are listed in Table 5.2.

Projector type	**Native resolution**	**Latency**	**Comments**
3M DMS 700	1024×768 @ 60 Hz	100 ms	short throw
Canon LV-S1E	800×600 @ 60 Hz	16.67 ms or less	office projector
Casio XJ-S30	1024×768 @ 60 Hz	16.67 ms or less	office projector
Epson EMP-400W	1280×800 @ 60 Hz	-16.67 ms	short throw
NEC WT610	1024×768 @ 60 Hz	16.67 ms or less	short throw
Sanyo PLC-XL50	1024×768 @ 60 Hz	16.67 ms or less	short throw
Sharp PG-A10X	1024×768 @ 60 Hz	16.67 ms or less	office projector
Toshiba EW25	1280×800 @ 60 Hz	16.67 ms or less	short throw

Table 5.2: Digital projector latency results.

Note that the Epson digital projector has a negative latency value. Benchmark results show that the Epson is almost one frame ahead compared to the CRT monitor.

5.1.4 Total system latency

It is now possible to calculate the total system latency. The results are displayed in Table 5.3. These test results do not include the application processing time.

Because of the 3M DMS 700 projector and the aforementioned Touchlib bug the total latency is almost 200 milliseconds.

In order to find out what influence latency has on the test results we selected a "small group" of test persons which did the test on different hardware. For this test session a patched version of Touchlib was used. The 3M DMS 700 projector was replaced by a Sharp PG-A10X projector. Because of the large throwing distance of the Sharp projector, a mirror was required for projection.

Used projector	**3M DMS 700**	**3M DMS 700** (improved touchlib)	**Sharp PG-A10X** (improved touchlib)
FireWire Camera	30 ms	30 ms	30 ms
Touchlib	33 ms	33 ms	33 ms
Touchlib ‘bug’	32 ms	0 ms	0 ms
Digital projector	100 ms	100 ms	0 ms
Total latency	195 ms	163 ms	65 ms

Table 5.3: Comparing the total system latency of different hardware and software combinations. Latency time is measured in milliseconds.

5.2 The experiment environment

The multi-touch tests are performed on a multi-touch table based on RI (as described in Section 2.6). To perform the test with the mouse the same computer was used. However in this case we used a CRT monitor (22 inch, Iiyama Vision Master Pro 510) and a mouse (Logitech standard mouse with three buttons). All test results are stored in a MySQL database.

5.2.1 The experiments conditions

To perform the measurements 22 persons were selected to participate. Each person is required to fill in a small question form which contains the following questions:

- Age
- Gender
- Handedness
- Using glasses?
- Educational background
- Number of years of experience with a mouse device
- Number of years of experience in general desktop computing
- Most used Operating System

Additionally each person was assigned a unique numerical identifier which is required to run all four experiments. All persons used the same order of doing the experiments. On completion of all experiments the test subjects are asked to fill out a small questionnaire (Appendix C).

In order to prevent a bias due to learning effects, the order of the used input devices for experiment 1 and 2 have been divided over the persons according to Table 5.4. The test persons were not paid for performing the experiments. The total time to complete all four experiments is about 25 minutes.

	Experiment 1		**Experiment 2**	
Group	**Device**	**Device**	**Device**	**Device**
1	Mouse	Multi-touch	Mouse	Multi-touch
2	Mouse	Multi-touch	Multi-touch	Mouse
3	Multi-touch	Mouse	Multi-touch	Mouse
4	Multi-touch	Mouse	Mouse	Multi-touch

Table 5.4: Four different experiment paths.

The experiments were performed by 22 users, 4 female and 18 male users. Most participants have a Bachelor or Master degree in Computer Science. Two participants were left handed and eight of them used glasses. All users have at least 7 years of experience in general desktop computing and using the mouse as input device. Fourteen users preferred MS Windows as operating system, 4 users Linux and 4 users Apple Mac OS.

5.3 Experiment 1: A 1D and 2D pointing task

5.3.1 Background

In 1954 Paul Fitts [11] of Ohio state University published 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 (Figure 5.2). The experiments were performed on different sizes of the metal plates (the width, W) and different distances between the two metal plates (the amplitude, A). During the test the apparatus recorded the time difference between touching the two metal plates. In order to measure the accuracy, Fitts added two extra plates on both sides of the metal plates. This way it became possible to register and record a miss.

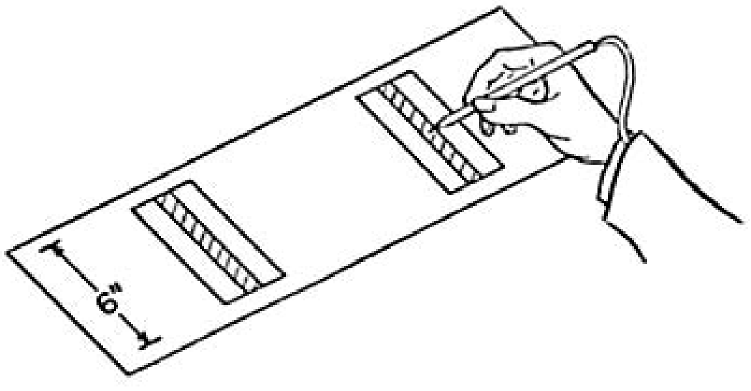


Figure 5.2: Reciprocal tapping apparatus (image take from [11]).

Fitts formulated the following equation to predict the movement time:

$$MT = a + b\log_2(\frac{2A}{W}) \tag{5.1}$$

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.

The logarithm part in Fitts' law describes how difficult the task is and is called the index of difficulty (ID). The unit of the ID is bits, as shown in the following equation:

$$ID = \log_2(\frac{2A}{W}).$$

$$MT = a + b \times ID \tag{5.2}$$

By taking a closer look at Equation 5.2 it can be separated in two parts. The first part contains a constant a that describes the cost of time that is spent to non-physical movement such as the processing time of recognizing objects. The second part multiplies the ID with a time unit which is often measured in milliseconds.

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 visible in Equation 5.4 or Equation 5.3:

$$IP = \frac{1}{b}, \tag{5.3}$$

$$IP = \frac{\overline{ID}}{\overline{MT}}. \tag{5.4}$$

When using the Equation 5.4, the impact of constant a is ignored. The performed measurements will use the first equation.

In some occasions equations 5.1 gives a negative rating for a task difficulty. This occurs when the amplitude is less than half the target width. To solve this issue MacKenzie [22] proposed the equation known as the Shannon formulation:

$$MT = a + b\log_2(\frac{A}{W} + 1), \tag{5.5}$$

$$ID = \log_2(\frac{A}{W} + 1). \tag{5.6}$$

By using the Shannon formulation we preserve the characteristics of Fitts' Law and are always presented a result with a positive rating.

5.3.2 Task description

For this experiment a Fitts' law model has been re-created in a modern digital environment using OpenGL. The application can be controlled using standard input devices such as a mouse, trackball, table or by using the multi-touch device.

Test A

In this first test two rectangular targets (one blue and one green) are presented to the test subject. The two rectangles appear on predefined locations (based on the amplitude) and widths from a data set, in a random order (see Figure 5.3.a). The test person is instructed to click or touch the green rectangles as fast as possible, but still keeping in mind that accuracy is more important than speed.

The data set contained 4 different values for width of the rectangle: 20, 30, 40 and 50 pixels, and 5 different amplitudes: 64, 128, 256, 384 and 512 pixels. In total, the data set contained 20 combinations. The most simple task has an ID of 1.19 bits (W = 50, A = 64) and the most difficult task has an ID of 4.73 bits (W = 20, A = 512).

Test B

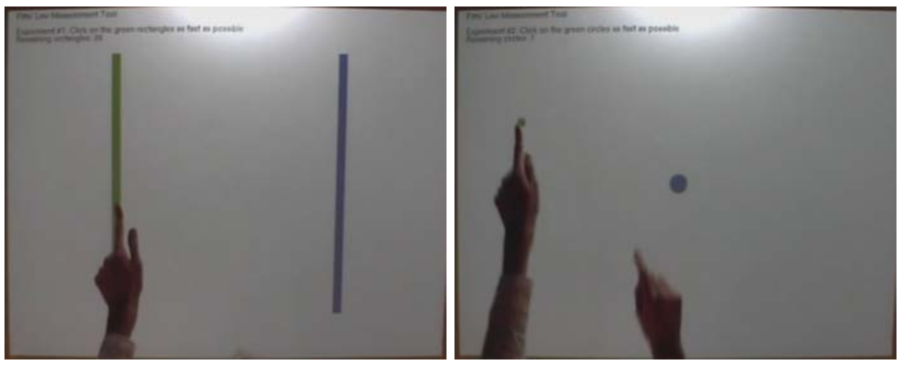
The second test is based on a model developed by MacKenzie that extends the current one dimensional testing model to two dimensions. The application is controlled in the same way as Test A, however, instead of rectangular shaped objects, circular objects are presented (see Figure 5.3.b). The first circle will always appear in the center of the screen functioning as a reference point. The second circle will appear on predefined locations using a predefined radius, in a random order. As in our previous test, the test subject is instructed only to click or touch the green objects.

The data set for Test B uses the same amplitudes and widths as Test A. The positions are divided in the two dimensional space.

During the test, the application records the time and position of the clicks between the two objects. When the experiment ends it calculates the ID and IP. To calculate the constants a and b, linear regression is performed on the data set of the measurements.

5.3.3 Training

Before performing the test run, all subjects are allowed to try out the test application on different input devices. During the training session all subjects are informed on how to use the multi-touch device and how sensitive the surface is.



(a) Test A (b) Test B

Figure 5.3: The left image shows a one dimensional Fitts' Law test. The right image shows a two dimensional Fitts' Law test.

5.3.4 Results

Before linear regression was used, outliers were removed from the data set. A point was considered an outlier when the value was more than one standard deviation from the mean. Detailed results with scatter plots can be found in Appendix D.

Results Test A

Results of Test A are shown in Figure 5.4. By taking a look at the intercept of the models, it shows that the mouse device has the lowest start cost. When the task difficulty increases (index of difficulty) the time to complete the task increases more compared to the multi-touch devices. The multi-touch devices show a higher start cost. When comparing the devices using the index of performance (Figure 5.5) of the input devices, the multi-touch devices performs better than the mouse in this task.

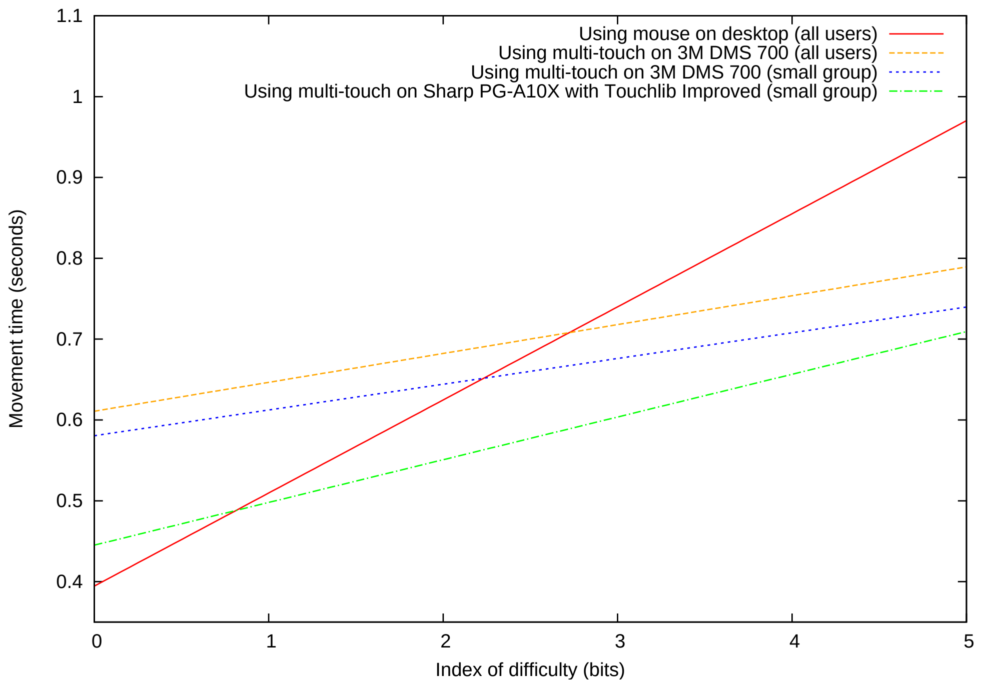


Figure 5.4: Comparison of the time to complete a task, based on the index of difficulty, on different input devices.

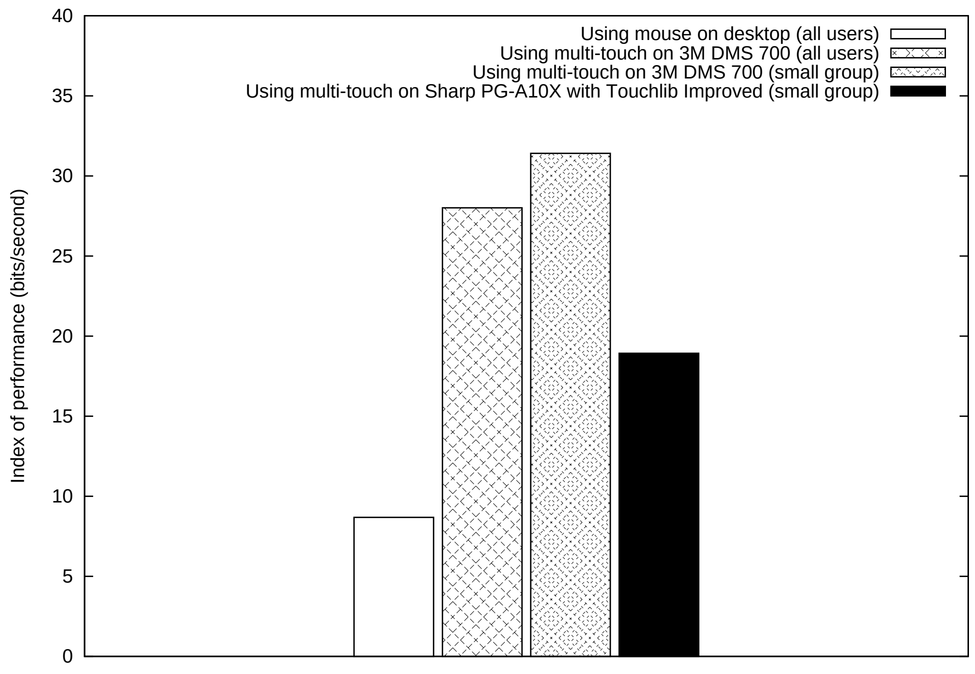


Figure 5.5: Comparison of the index of performance of four different input devices.

In Table 5.5 we present the results of the input devices including Fitts' Model, the index of performance and the linear model fit. The table shows that the measurements of the mouse device contain a high correlation ($r = 0.812$). On the multi-touch device, results show that the measurements using the 3M projector, contain a low correlation.

Input device	**Model**	**IP**	r	r^2
Mouse (Desktop) All users	0.39 + (0.12 * ID)	8.68	0.812	0.659
Multi-touch (3M) All users	0.61 + (0.04 * ID)	28.01	0.435	0.189
Multi-touch (3M) Small group	0.58 + (0.03 * ID)	31.41	0.370	0.137
Multi-touch (Sharp) Small group	0.45 + (0.05 * ID)	18.93	0.689	0.475

Table 5.5: A comparison between different input devices presenting Fitts' model, the index of performance (IP) and the model fit.

Results Test B

Compared to Test A, the slope of the input devices in Figure 5.6 shows different behavior. When increasing the task difficulty, it becomes more difficult to perform the task on the multi-touch table when the 3M projector is used. The difference between the slope of the mouse input device and the multi-touch input device using the Sharp projector is minimal.

By comparing the index of performance in Figure 5.7, it shows that the multi-touch table using the Sharp projector outperforms the mouse in this set of tasks.

In Table 5.6 we present the results of the input devices including Fitts' Model, the index of performance and the linear model fit. The table shows that the measurements of the mouse device contain a high correlation ($r = 0.897$). Compared to Test A, Test B shows higher values for the measurements correlations when using the multi-touch device.

5.3.5 Discussion

Results from Test A show that the multi-touch devices contain higher start cost than the mouse device. The reason for this can be related to the fact that it requires

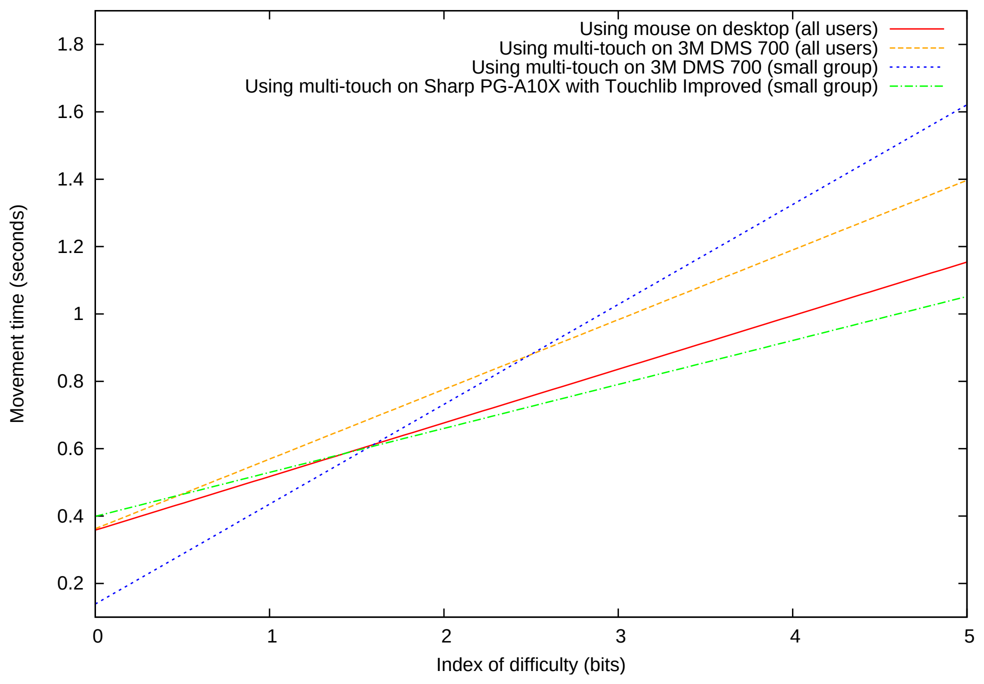


Figure 5.6: Comparison of the time to complete a task, based on the index of difficulty, on different input devices.

more physical effort to move a hand to a specified destination than using a mouse to move a cursor. The index of performance however, is lower when performing two dimensional tasks. The reason for the large difference between the IP in Test A and Test B when using the multi-touch table is the fact that the user is required to perform more physical effort. In Test A the movement of the hands was only restricted in one dimension. Because Test B contains situations where large and small objects are placed at the top of the screen, the user is required to reach further. An example can

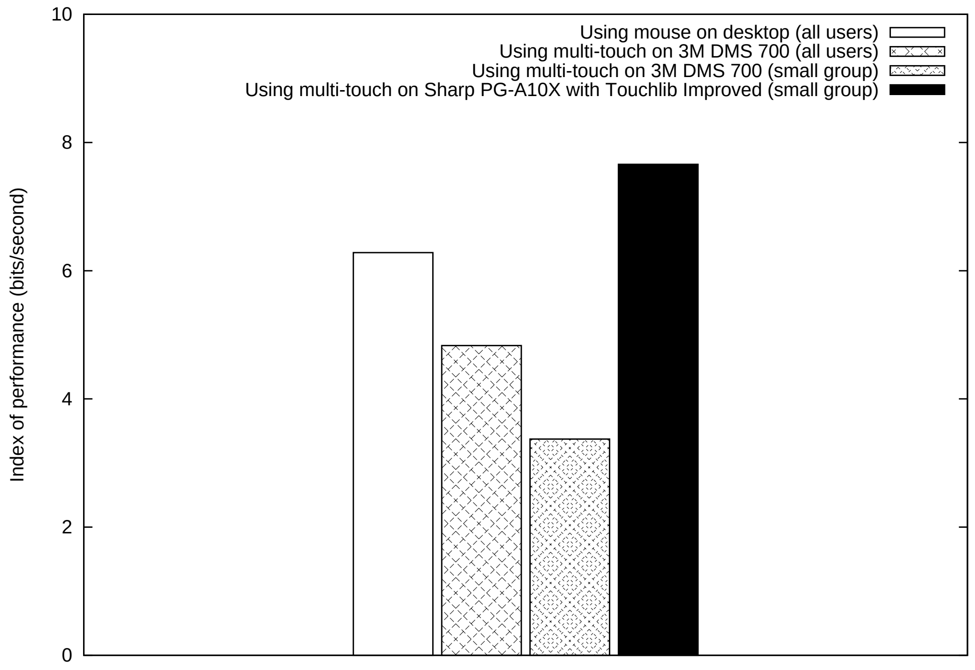


Figure 5.7: Comparison of the index of performance of four different input devices.

Input device	**Model**	**IP**	r	r^2
Mouse (Desktop) All users	0.36 + (0.16 * ID)	6.28	0.897	0.805
Multi-touch (3M) All users	0.36 + (0.21 * ID)	4.83	0.576	0.332
Multi-touch (3M) Small group	0.14 + (0.30 * ID)	3.37	0.621	0.385
Multi-touch (Sharp) Small group	0.40 + (0.13 * ID)	7.66	0.548	0.300

Table 5.6: A comparison between different input devices presenting Fitts' model, the index of performance (IP) and the model fit.

be found in Figure D.6, it shows that users have difficulty to select the object in the task with an ID of 4.73 bits. This corresponds to the task with the object width of 20 pixels (1.85 cm) and an amplitude of 512 pixels (47.35 cm).

5.4 Experiment 2: An object manipulation task

With the introduction of new multi-touch input device often object manipulation based applications are presented (such as photo manipulation). By using predefined sets of gestures the user can move, scale and rotate the object. The following experiments focuses on how multi-touch (gestural) input compares to mouse input on object manipulation.

5.4.1 Task description

Based on object manipulation test described in [12] we created a similar test application. In this test the user is required to manipulate a green square shaped object matching the position, orientation and size of the red square shaped object (Figure 5.8). At the start of each test run, the user is required to place the mouse pointer or finger at a home location. After placing the pointer at the home location, a countdown timer will start. When the timer reaches zero, the scene will display the green and red objects. The user is instructed to move the green object onto the red object. When the green object reaches its destination and the user releases press (or touch), the system compares the position of the red and green object. If the difference is lower than the tolerance, the object will snap and the test will continue to the next scene. During the test, the start time, object selection time, the end time and the number of selection and docking errors are recorded. A selection error occurs when a user misses the green object. A docking error occurs when a user releases the green object without matching the tolerance. The data set contains a combination of the following properties:

- 3 target sizes (100, 125 and 150 pixels),
- 3 target scales (0.5×, 1.0× and 2.0×),
- 3 target angles (90, 180 and 270 degrees),
- 3 target distances (256, 384 and 512 pixels).

We used the following tolerance settings:

- 4 pixels tolerance in x and y direction,
- 2 degrees in angle,
- 0.08× in scale.

The data set:

- Test A: 3 distances × 3 sizes = 9 combinations,
- Test B: 2 scales × 3 distances × 3 sizes = 18 combinations,
- Test C: 3 angles × 3 distances × 3 sizes = 27 combinations,

- Test D: 2 scales × 3 angles × 3 distances × 3 sizes = 54 combinations,
- Total combinations: 108.

In this data set duplicate entries were removed. Because of the large number of combination we divided the data set in three equally sized sets. When the experiment starts, the computer chooses a data set randomly.

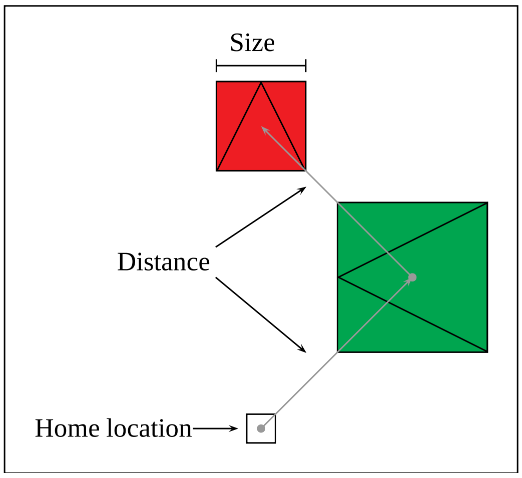


Figure 5.8: A scene of experiment 2. The home location is placed on the bottom. In this example the green object has a different rotation and scale than the red object. The user is required to adjust the green object to fit the red object.

Test A - Translate

In this first test the manipulation of the green object is restricted to translation. Rotation and scaling are disabled.

Test B - Translate and scale

This test extends the previous test with a different sized destination object. Rotation is disabled.

Test C - Translate and rotate

This test extends Test A with a differently orientated destination object. Scaling is disabled.

Test D - Translate, scale and rotate

In the final test of experiment 2 the destination object has a different size and rotation.

5.4.2 Training

Because multi-touch gestures are a new way of interaction for most test subjects, each will be briefly introduced:

When using the mouse the following controls are available:

- Translation: hold left mouse button + movement on the X-axis/Y-axis,
- Scaling: hold middle mouse button + movement on the Y-axis,
- Rotation: hold right mouse button + movement on the X-axis.

When using the multi-touch table the following gestures are available:

- Translation: touch the surface + movement on the X-axis/Y-axis,
- Scaling: touch the surface with two fingers, the delta of the distance is used to zoom in or out,
- Rotation: touch the surface with two fingers and rotate the current position.

Each person is allowed to practice the interaction methods for a few minutes.

5.4.3 Results

Due to a typo when creating the data set, several test entries that should contain the distance value of 256 pixels were assigned a distance value of 281 pixels. Unfortunately it was not possible to redo the experiments.

In the experiment results we define the test conditions with a Test ID. The Test ID is a string such as: 150w-000a-100s-256d-R. This represents the Width (w), the rotation Angle (a), the Scale (s), the travel Distance (d) and start location (Left or Right) of the object. The results of Test A are shown in Figure 5.9. The bar chart displays the average time to complete a specific task. Results show that the multi-touch devices require more time to complete the tasks of Test A. Results of the multi-touch device using the 3M projector show that the docking time is about 2 to 3 times slower than when a mouse is used.

In Figure 5.10 a comparison of task difficulty is made based on the different scales and rotation. For each device we compared the total time to complete the task (selection and docking). Results of the mouse device show minimal differences between the time to complete a task containing scaling or rotation. The results of the multi-touch device using the 3M projector show that more time is required to complete a task that contains a rotation than a task that contains scaling.

To compare the performance of gesture based interaction on the multi-touch device with the mouse, the test results have been sorted by task difficulty according to the completion time when using the mouse device (Figure 5.11 and Figure 5.12).

The number of selection errors on all devices was either zero or near zero. Therefore no graph is presented for this result. The results of the number of docking errors is shown in Figure 5.13. In Test A results of the mouse device show a result of zero or near zero. This means that after object selection, the object is dragged to the location within the tolerance in one stroke. The multi-touch devices however show that multiple actions are required in order to dock the object. When comparing Figure 5.12 with Figure 5.14 difficult mouse tasks show an increasing error rate for docking.

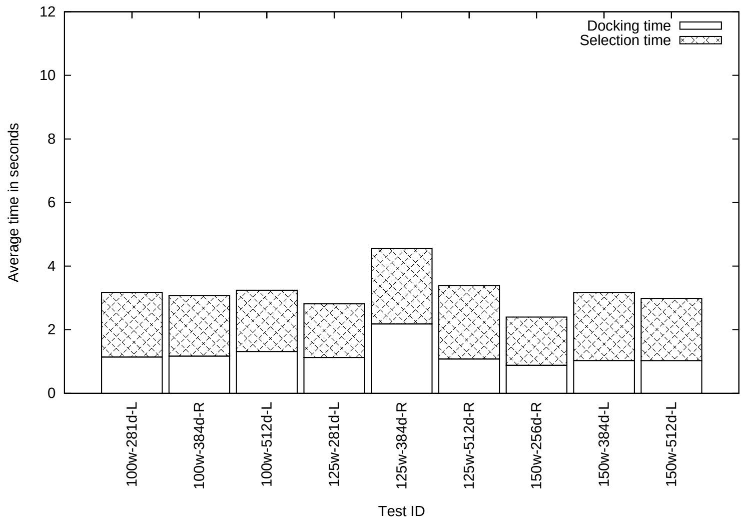
5.4.4 Discussion

When comparing the results of the input devices of Test A, it is visible that most time is spent on docking the object. The selection time on each device does not seem to be influenced by the size and travel distance between the home location and the object.

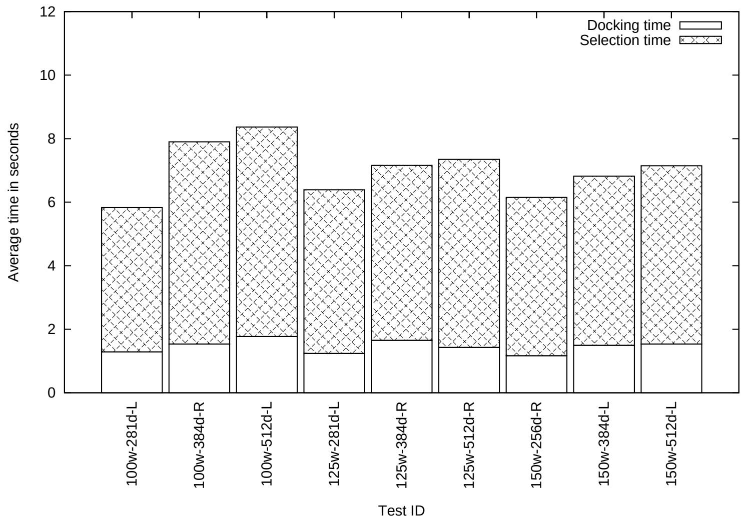
Results show that in Test B and Test C, rotation on a multi-touch device requires more time than scaling. This can be explained by the fact that when scaling is performed, two points of contact change in distance. When the task contains a rotation, more physical effort is required. Due to the limitations of the joints in our arms and hands, motions that require a rotation of 180 degrees are often performed in multiple steps.

When comparing the total completion time of the task in Test A, results show that a mouse is more suited for point and dragging tasks. Results of Test B and C show that when the task difficulty of the mouse increases, gesture based interaction on the multi-touch device (Sharp) is able to perform the task faster. The result of Test D show that when the complexity of tasks increases (scaling and rotation), the multi-touch device (Sharp) outperforms the mouse device on the most time consuming tasks.

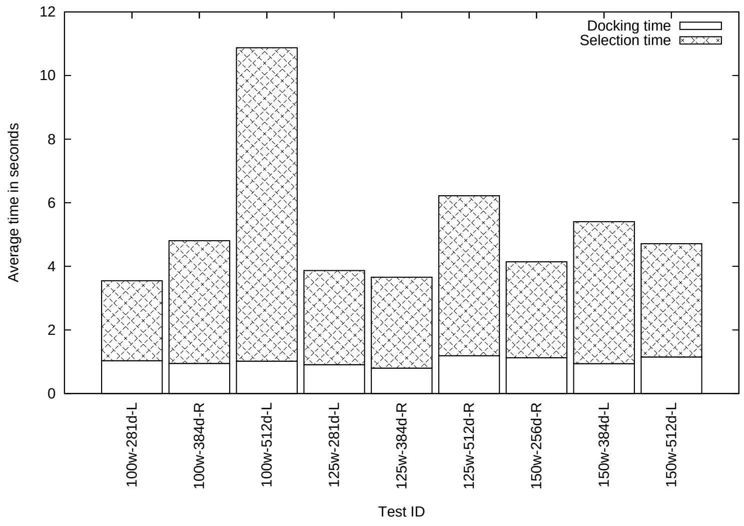
By comparing the results of Test B and C on docking errors, we notice that more actions were required to complete a task containing rotation than scaling. This explains why more time was required to complete the task with rotation in Figure 5.10.



(a) Mouse input device

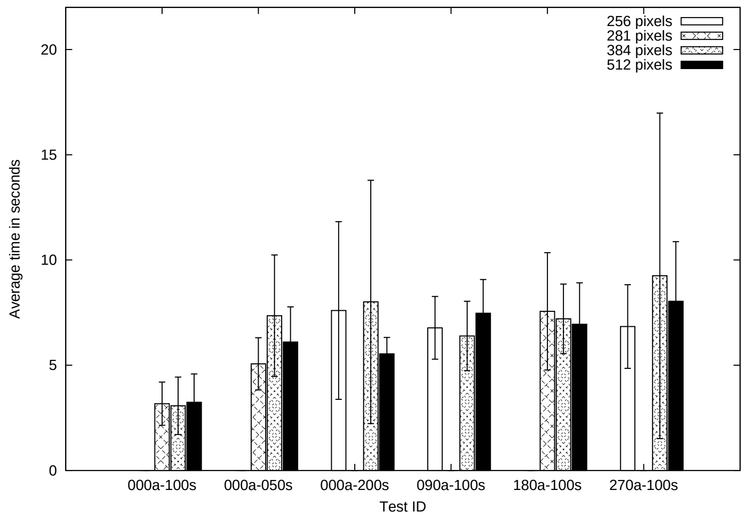


(b) Multi-touch input device (3M DMS 700)

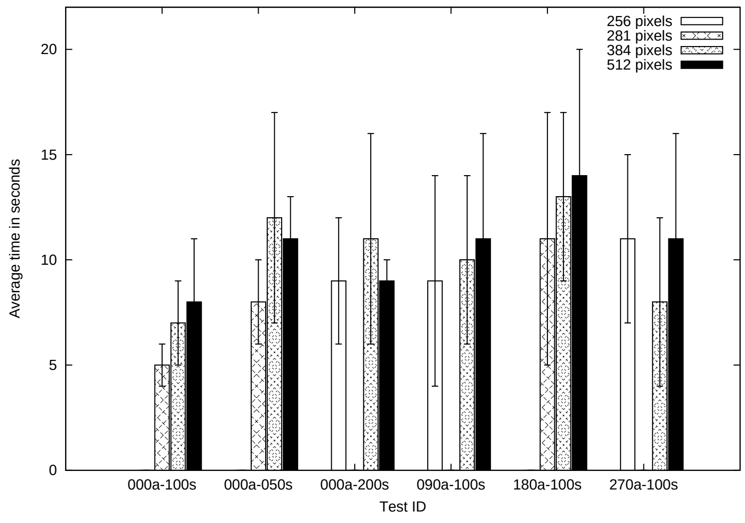


(c) Multi-touch input device (Sharp PG-A10X)

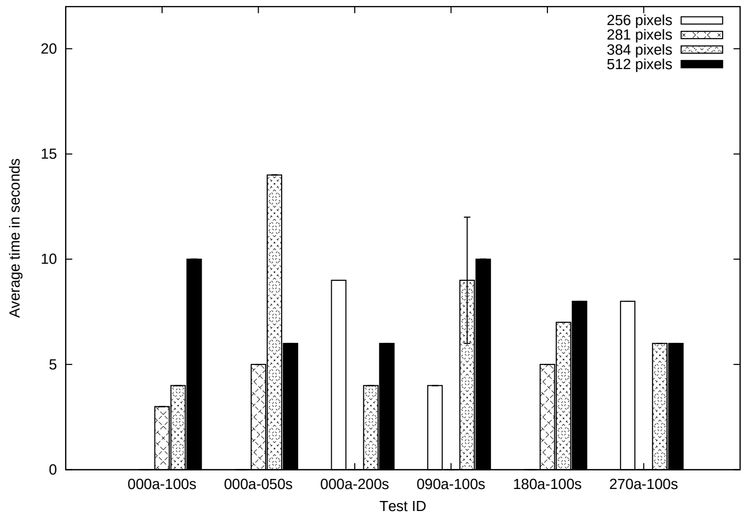
Figure 5.9: A comparison of the time to complete tasks from Test A. The histogram is divided into the time required to select the object and the time to dock the object. The x-axis specifies which test is performed using a Test ID. The Test ID consists of: Width (w), Distance (d) and start location (Left or Right) of the movable object.



(a) Mouse input device

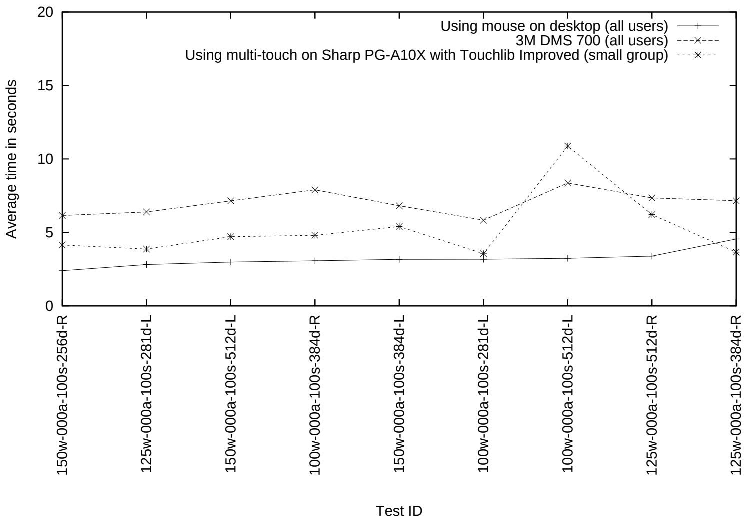


(b) Multi-touch input device (3M DMS 700)

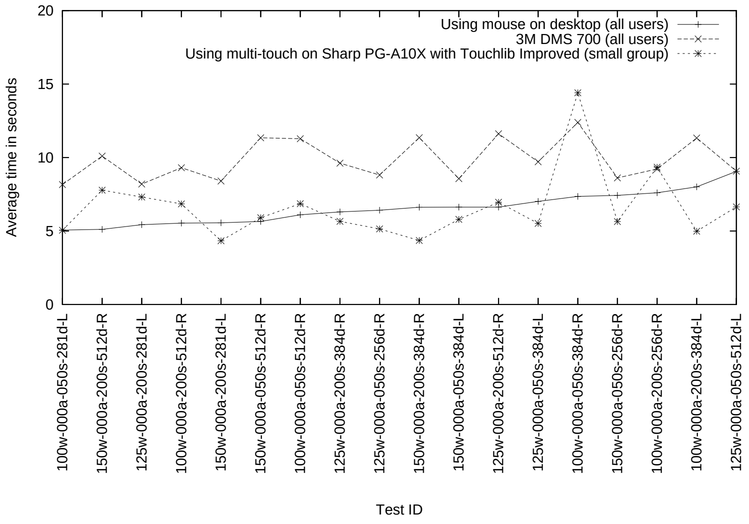


(c) Multi-touch input device (Sharp PG-A10X)

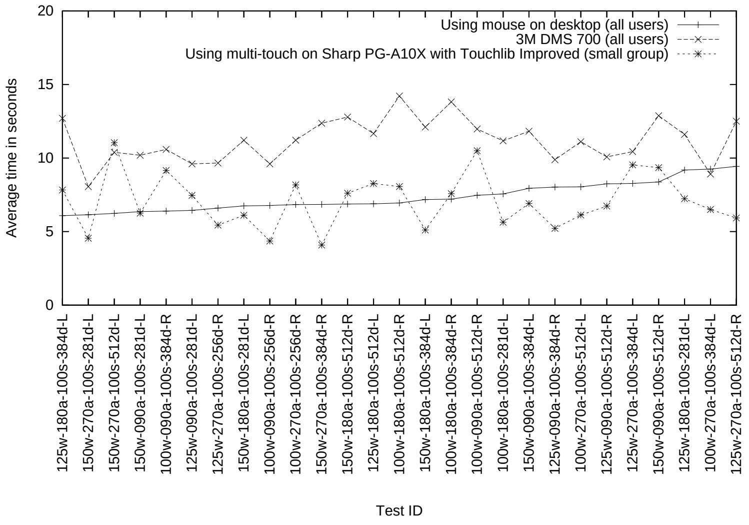
Figure 5.10: A comparison of task difficulty of Test B (using different scales and rotation). In this data set objects are 100 pixels wide. Each bar represent a different value for the travel distance. The y-axis presents the time required to complete the task, the x-axis specifies which test is performed using a Test ID. The Test ID consists of the Angle (a) and Scale (s).



(a) Test A



(b) Test B



(c) Test C

Figure 5.11: A comparison of task difficulty sorted by the time it takes to complete a task with the mouse device. The y-axis specifies the time it is required to complete the task. The x-axis specifies the Test ID. The Test ID consists of: Width (w), Angle (a), Scale (s), Distance (d) and start location (Left or Right) of the movable object.

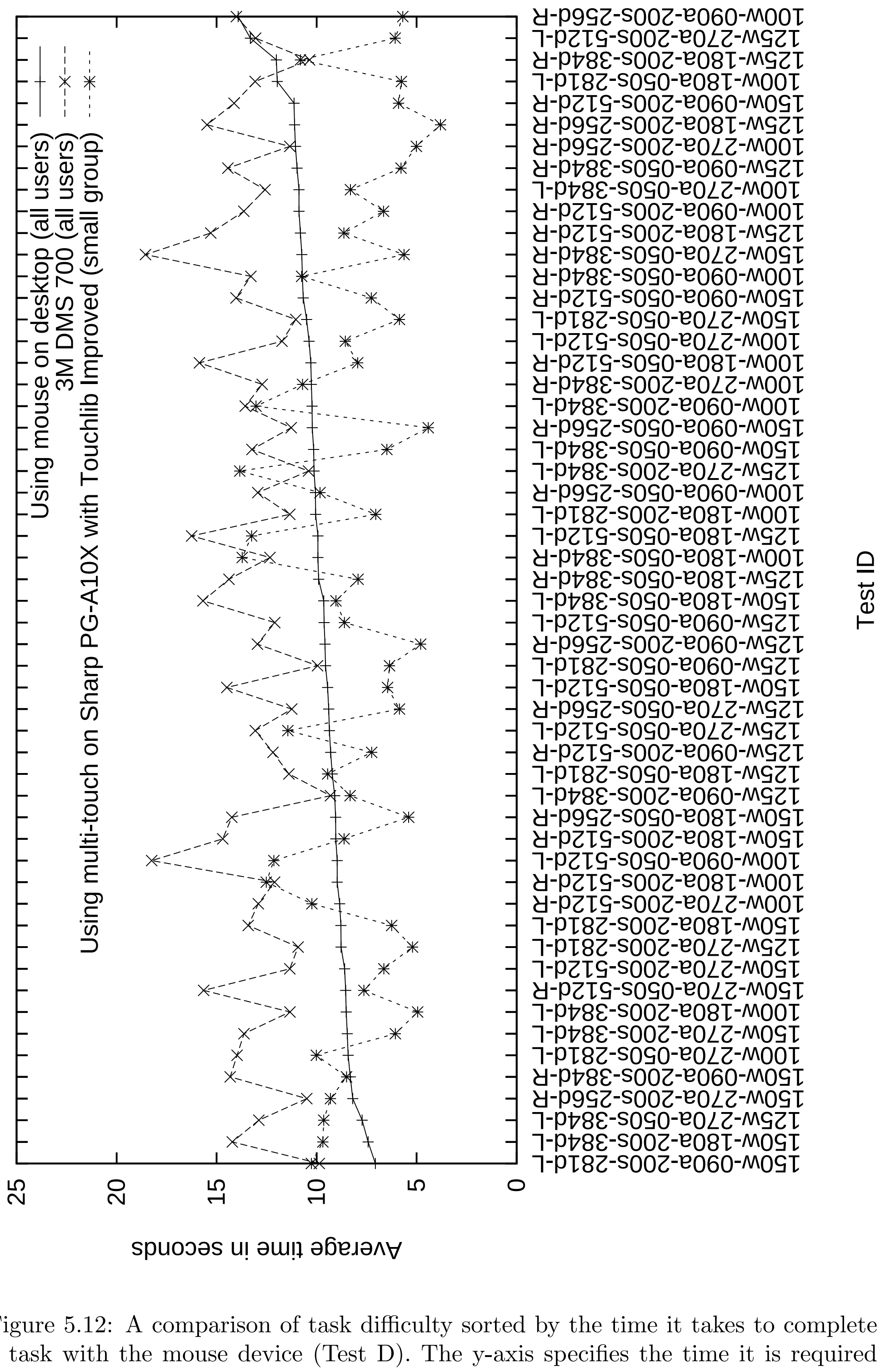
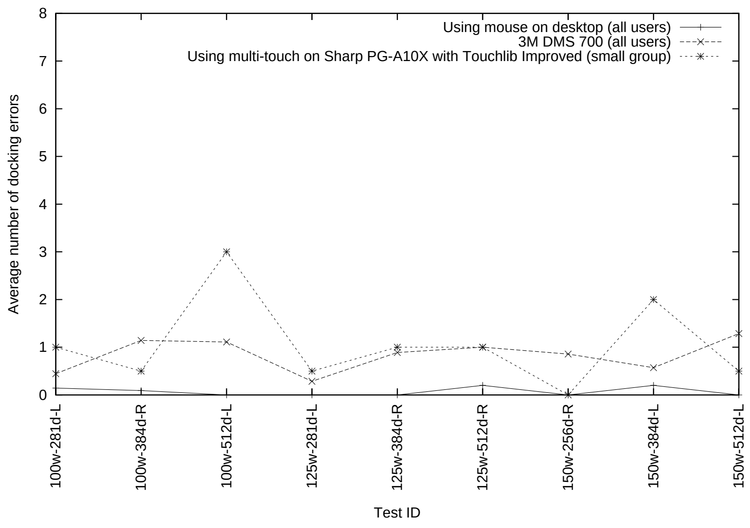
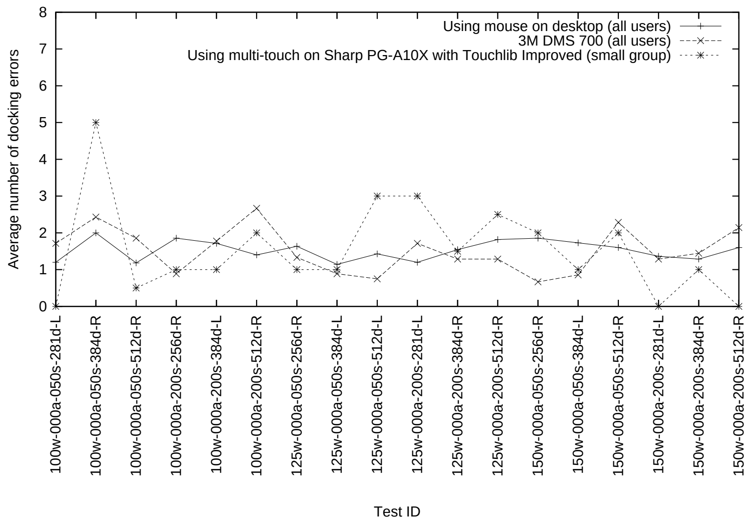


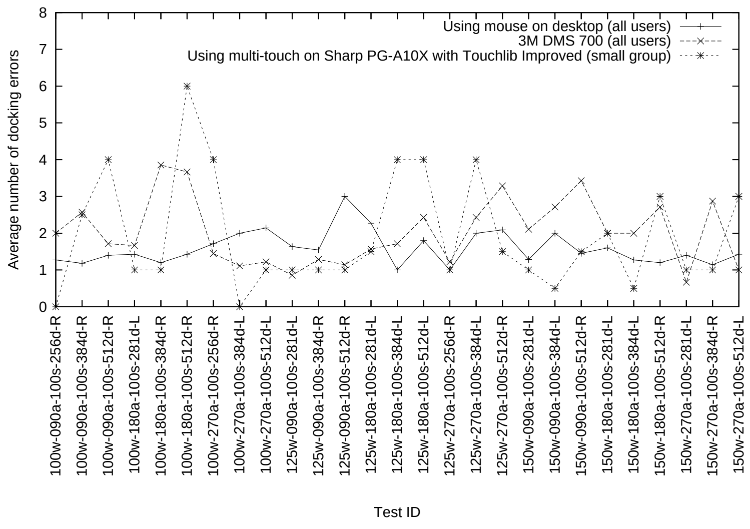
Figure 5.12: A comparison of task difficulty sorted by the time it takes to complete a task with the mouse device (Test D). The y-axis specifies the time it is required to complete the task. The x-axis specifies the Test ID. The Test ID consists of: Width (w), Angle (a), Scale (s), Distance (d) and start location (Left or Right) of the movable object.



(a) Test A



(b) Test B



(c) Test C

Figure 5.13: A comparison of the number of docking errors when comparing task difficult based on translation, scaling and rotation. The y-axis specifies the time it is required to complete the task. The x-axis specifies the Test ID. The Test ID consists of: Width (w), Angle (a), Scale (s), Distance (d) and start location (Left or Right) of the movable object.

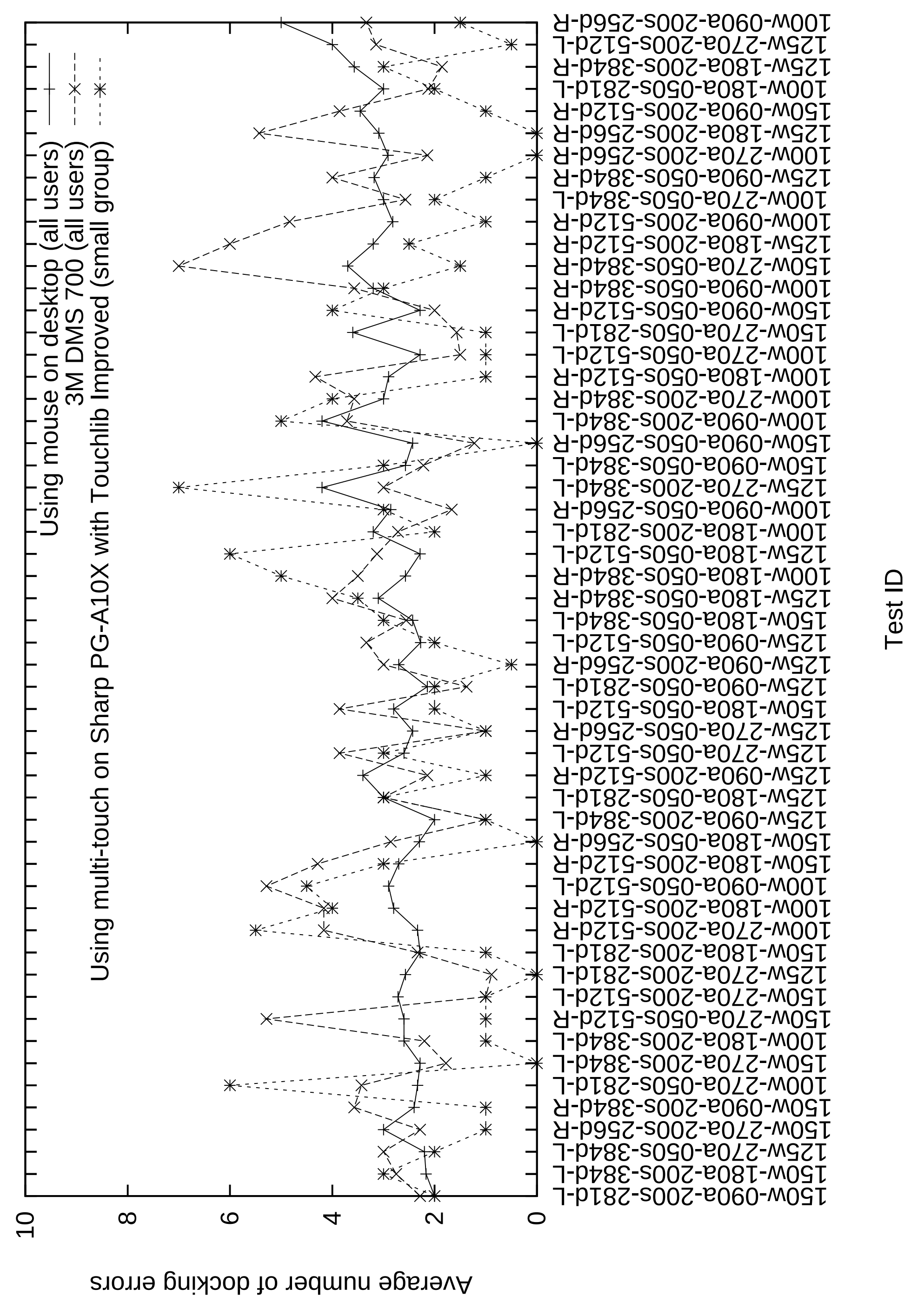


Figure 5.14: A comparison of the number of docking errors compared to the task difficulty sorted by the time it takes to complete a task with the mouse device. The x-axis specifies the Test ID. The Test ID consists of: Width (w), Angle (a), Scale (s), Distance (d) and start location (Left or Right) of the movable object.

5.5 Experiment 3: A collaborative sorting task

5.5.1 Introduction

Large interactive tabletop displays allow multiple people to perform tasks collaboratively. Whilst our current hardware does not allow us to distinguish between the users (unlike the Mitsubishi DiamondTouch) it does allow multiple users to perform a set of tasks. In our third experiment we focus on collaborative tasks in small groups which consist of four persons maximum.

5.5.2 Task description

For this experiment a tile based application was created. The task is to sort colored square shaped objects to their corresponding colored containers (40 objects in total). The four containers are placed in the middle of the screen. The task is performed with different numbers of test subjects (one to four). The test subjects are encouraged to sort quickly, but instructed that the emphasis should be on accuracy. In the test it is possible to put objects in the wrong container. During the test, the number of test subjects, the task completion time, parallel activity and the number of mistakes is recorded.

5.5.3 Training

Before the test, all subjects are introduced to the application and allowed to try the application together for a few minutes.

5.5.4 Results

The results of the collaborative sorting task are presented in Figure 5.15. The figure shows that the number of users has a positive influence on the time to complete the task.

When comparing the results using the 3M projector, the figure shows that two users are able to complete the task at 71% of the original completion time (small group). When three users complete the task, the time to complete the task is 49% of the original time. Completing the task with four users does not seem to improve performance.

Collaboration on a Sharp projector showed better improvements than on the 3M projector. We measured that two users are able to complete the task at 42% of the time when completing the task alone (small group). When completing the task with three users, minimal improvement is noticeable. By using four users the time to complete the task is reduced to 27% of the time.

The results in Figure 5.16 show that the number of strokes increases when more users are active. According to Figure 5.17 and Figure 5.18 increasing the number of users also increases the number of selection errors and sorting mistakes.

Results of the parallel activity measurements are presented in Figure 5.19.

5.5.5 Discussion

When increasing the number of users, the number of selection errors and strokes increases. This indicates that the system is either losing track of the user's input or users are blocked by movements of other users. Results from Figure 5.19 show that on average, users are able to move three tiles at the same time.

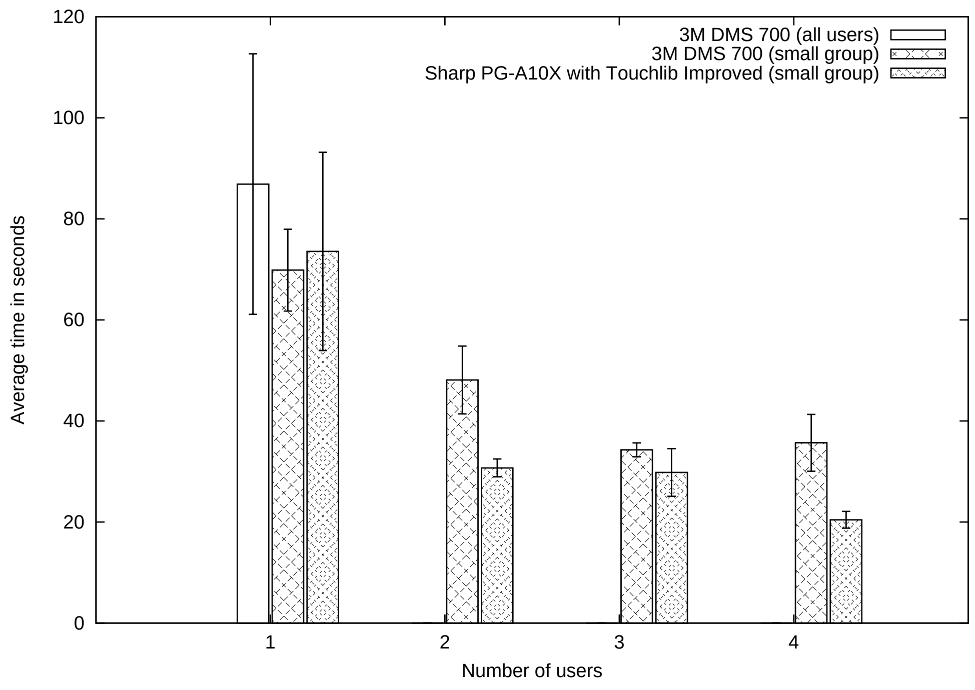


Figure 5.15: Average time in seconds to complete the sorting task.

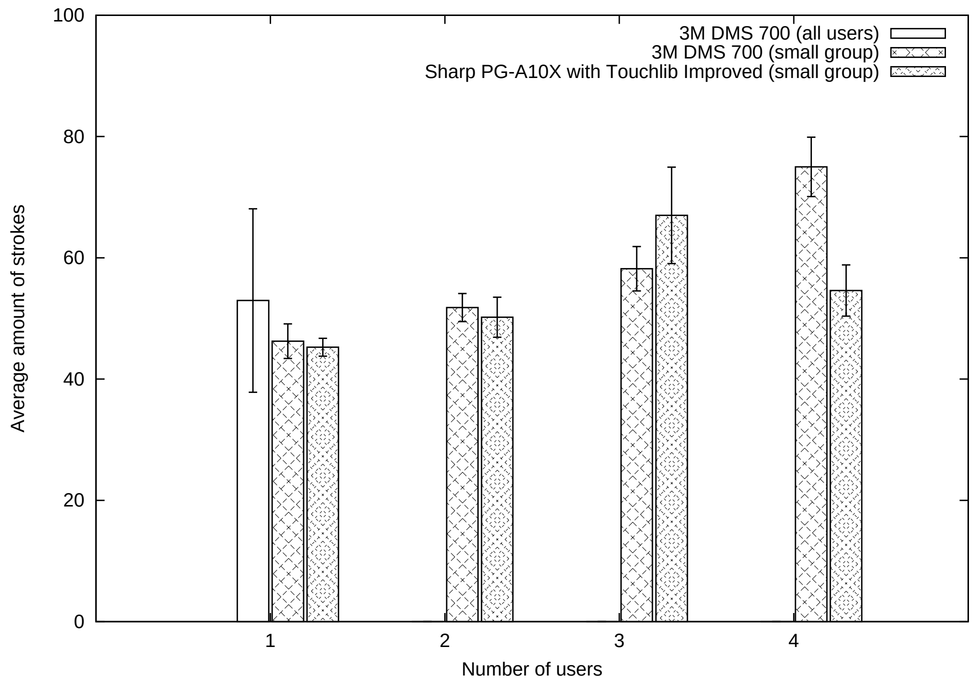


Figure 5.16: Average number of strokes used to complete the sorting task.

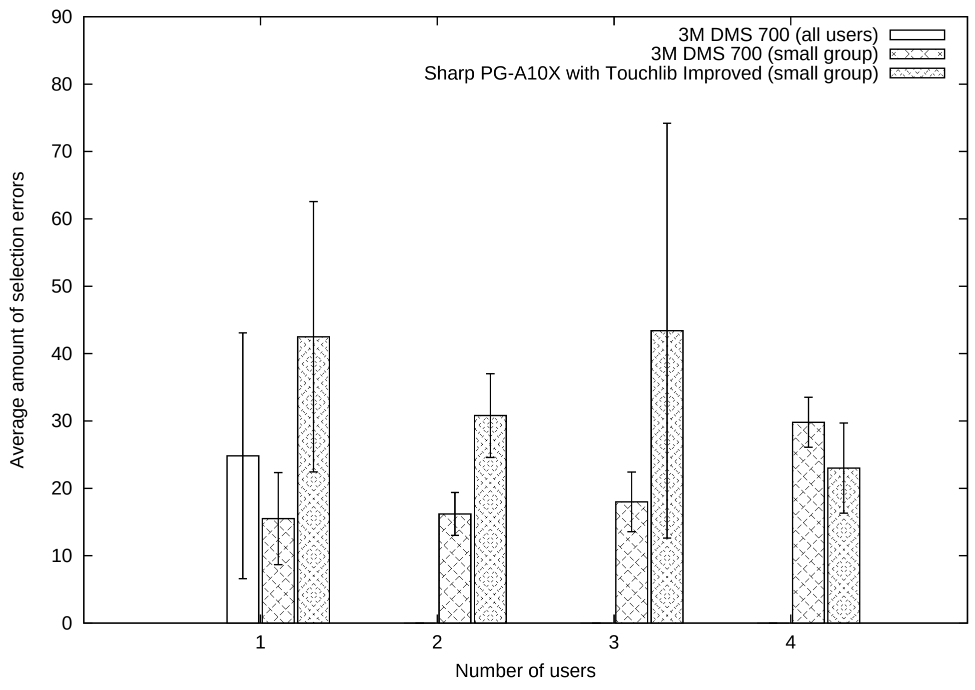


Figure 5.17: Average number of selection errors when completing the sorting task.

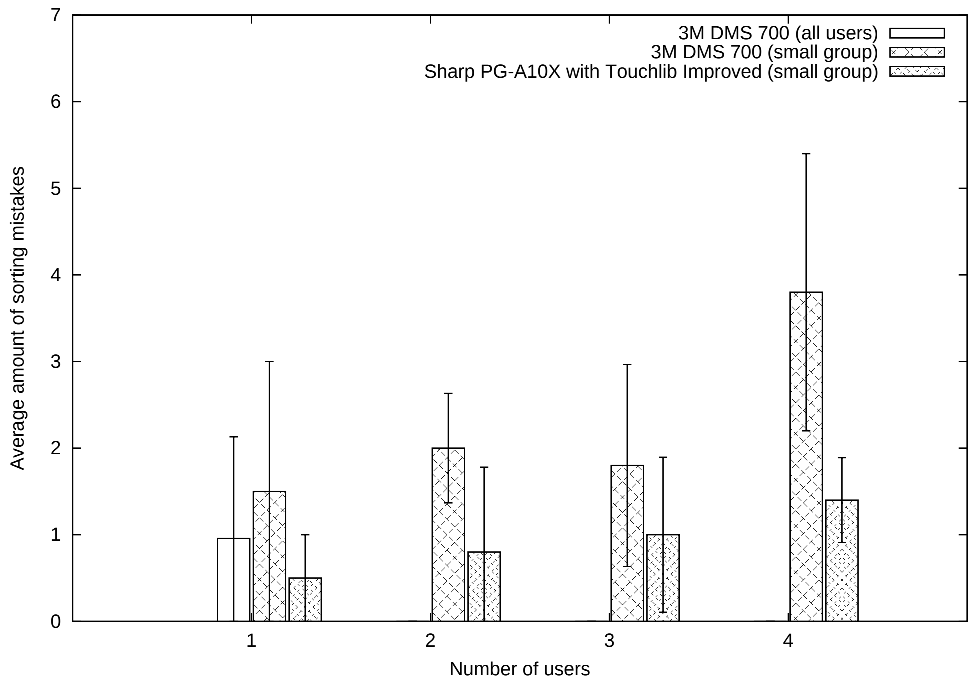


Figure 5.18: Average number of sorting mistakes when completing the sorting task.

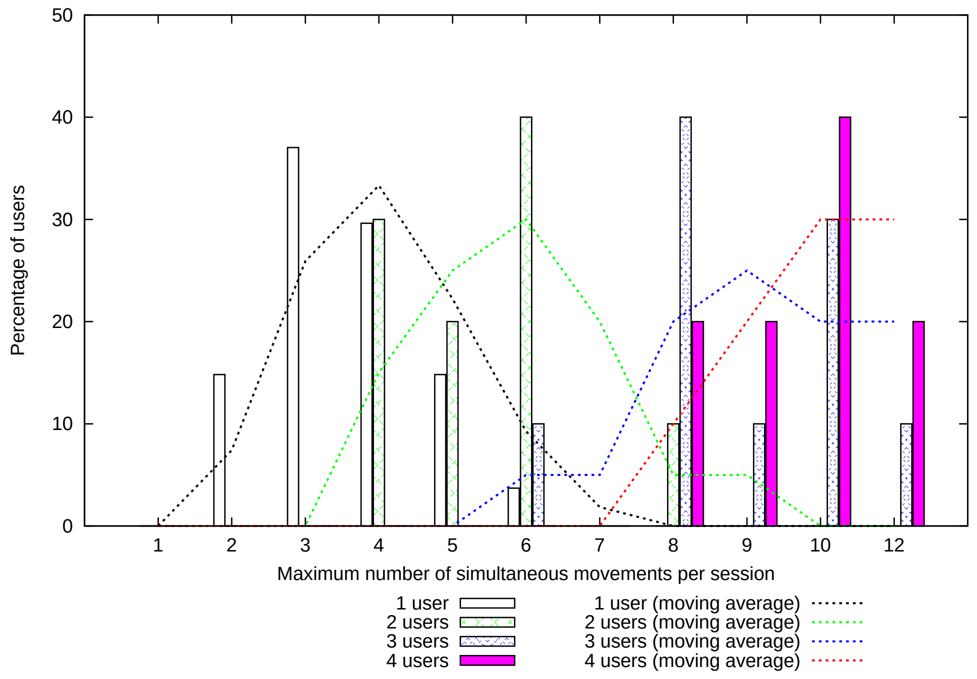


Figure 5.19: A comparison of parallel activity with different number of users.

5.6 Experiment 4: A collaborative point and selecting task

5.6.1 Introduction

Our previous collaborative task focused on dragging objects in a two dimensional space. Another interesting task is how well users can collaborate in point and select tasks. In our final experiment we focus on collaborative tasks in small groups which consist out of four persons maximum.

5.6.2 Task description

Test A - Pointing

For this experiment the screen is filled with different colored objects (50 objects, 5 colors). The task is to touch the objects in the right order. When touched the object disappears. During the test the order is displayed at the top of the screen. The experiment ends when all of the requested objects are removed.

Test B - Selecting

This experiment is based on Test A, however instead of pointing at objects we allow the user(s) to select one or multiple objects at the same time to clear the scene. Selecting is done by drawing a closed line over the objects ("rubber band" selection). Again, the experiment ends when objects are removed.

All tests are performed with different group sizes (one up to four). During the test the number of test subjects, the task completion time and the number of mistakes is recorded.

5.6.3 Training

Before testing all test subjects are introduced to the application and allowed to try the application together for a few minutes.

5.6.4 Results pointing task

Results from the collaborative pointing task (Figure 5.20) show an improvement in time when multiple users are active. Comparing the result when using the 3M projector shows improvements in completion time reduced to 58% (2 users), 35% (3 users) and 39% (4 users). When using the Sharp projector the difference between two, three and four users is less noticeable. The completion time is reduced to 46% (2 users), 41% (3 users) and 39% (4 users). Results are compared to the small group test results.

The number of selection and color errors is increased with the number of users (Figure 5.21 and Figure 5.22).

5.6.5 Results selection task

Results from the collaborative selection task can be found in Figure 5.23. When completing the task on with the 3M projector the completion time is reduced to 48% (2 users), 37% (3 users), 37% (4 users). The improvements on the Sharp projector are: 42% (2 users), 39% (3 users) and 33% (4 users). The difference between three and four active users is minimal.

According to Figure 5.24 and Figure 5.25 selection errors and color mistakes are higher when more users are active.

5.6.6 Discussion

In the pointing task we noticed from observation that when more users are active on the table, they are more likely to block others when selecting objects, this has a negative impact on the time to complete the task.

The test results of the selection task show minor improvements between three and four active users. The reason for this can be found in the design of the experiment. Since users are required to select objects in the right order by dragging a closed line

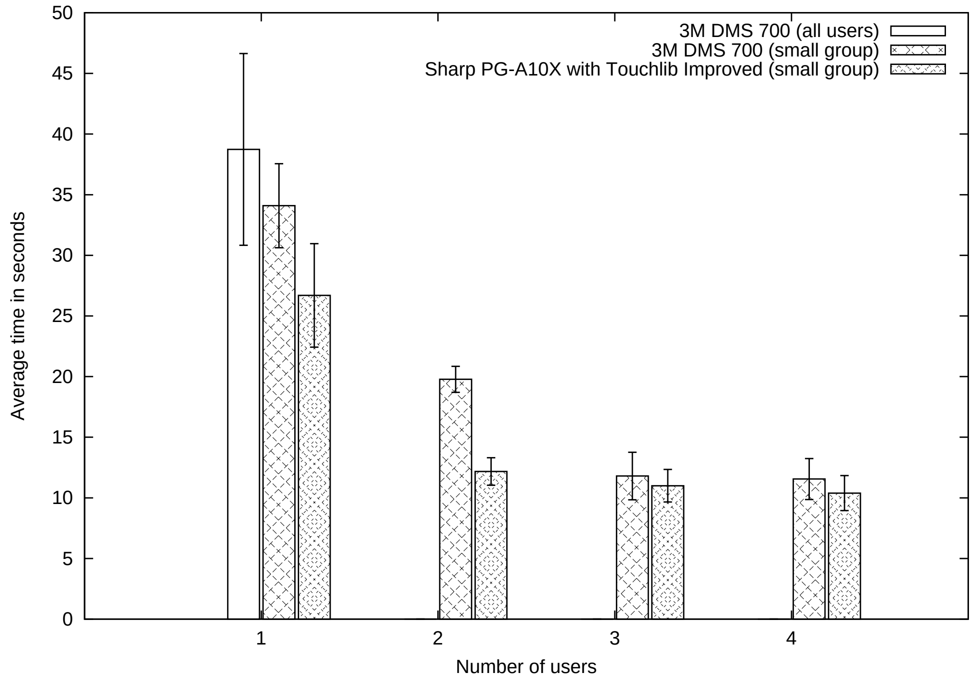


Figure 5.20: Average time to complete the pointing task.

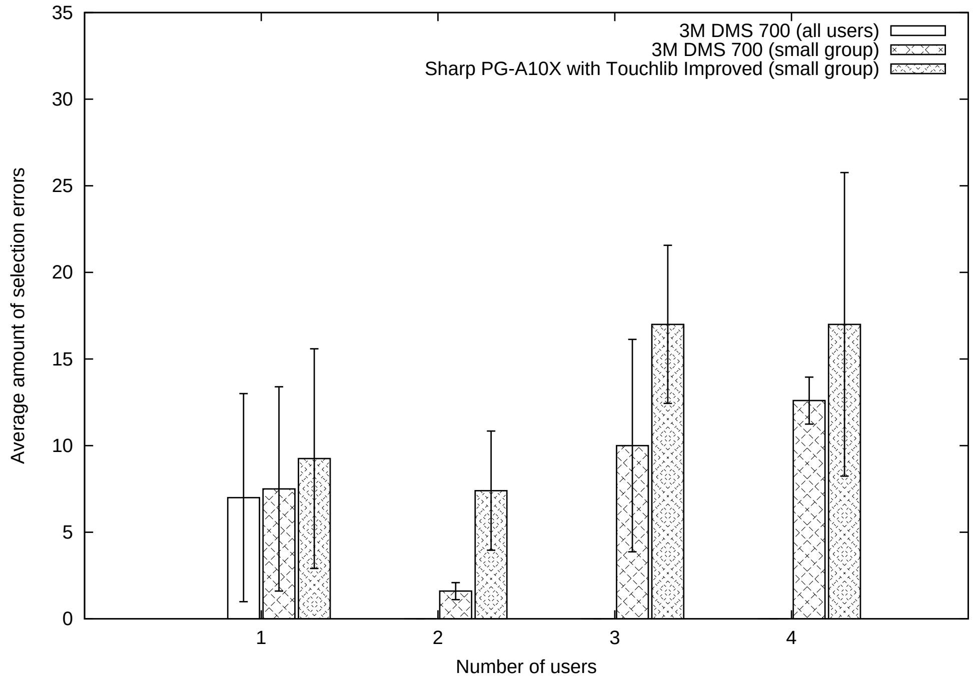


Figure 5.21: Average number of selection errors when completing the pointing task.

over the objects, it is more likely that users need to wait before they can proceed to the next object color.

Compared to the pointing task, the selection task has a higher selection error rate. From observation we noticed that the corners of the table are less responsive to touch. Therefore, it often requires multiple trials to clear objects near the corners of the table resulting in a high number of selection errors.

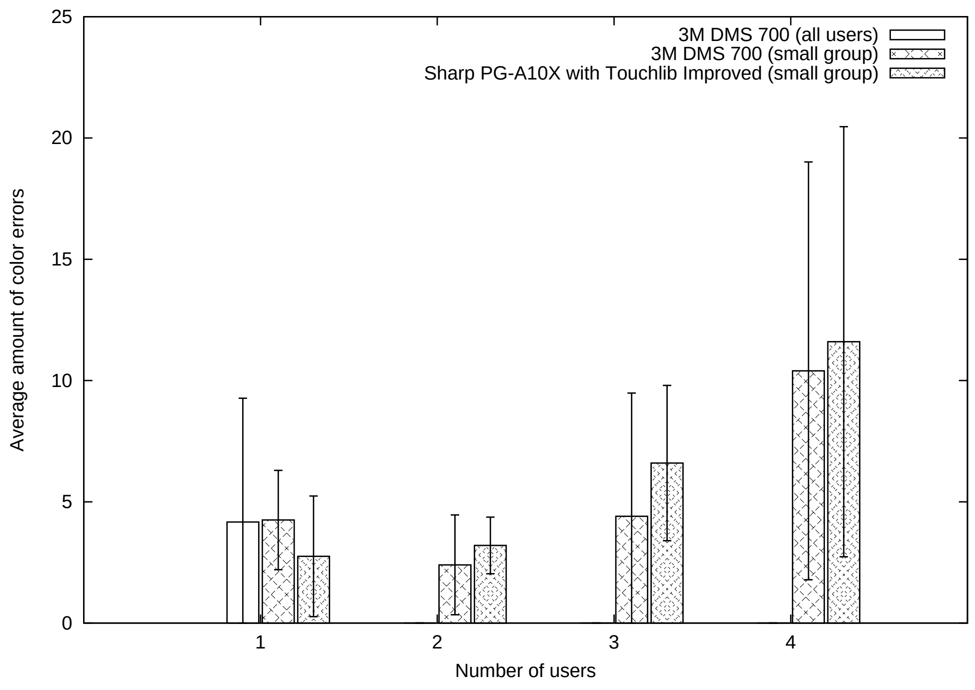


Figure 5.22: Average number of color mistakes when completing the pointing task.

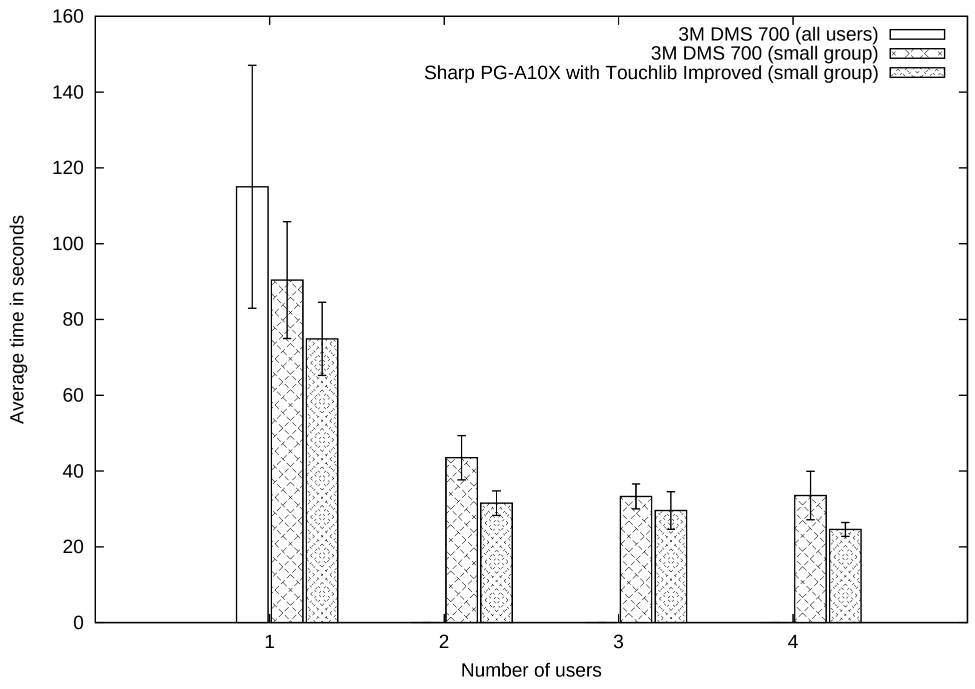


Figure 5.23: Average time to complete the selection task.

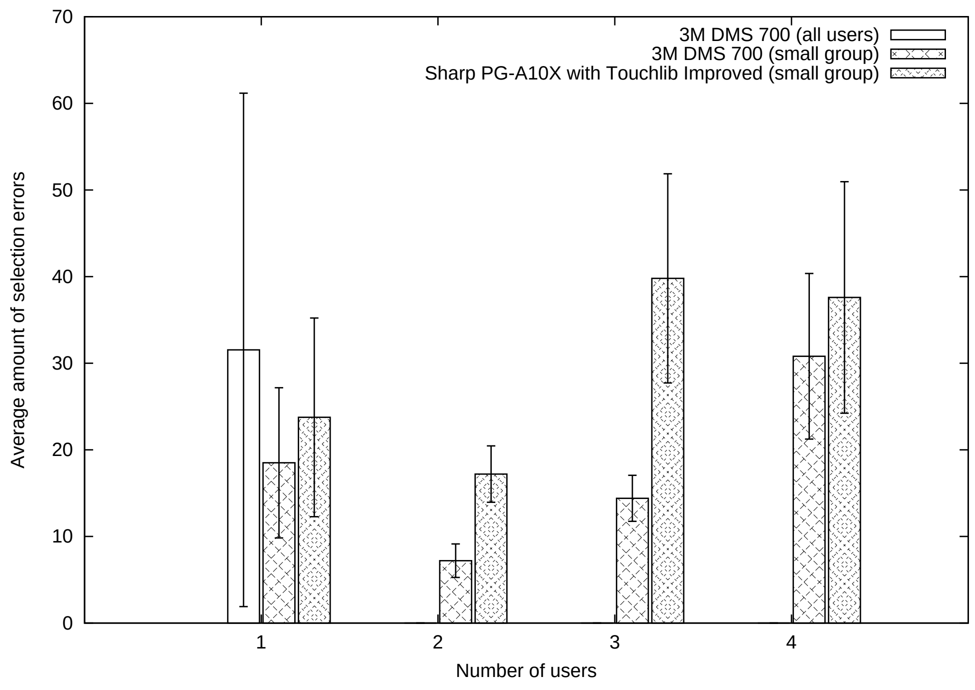


Figure 5.24: Average number of selection errors when completing the selection task.

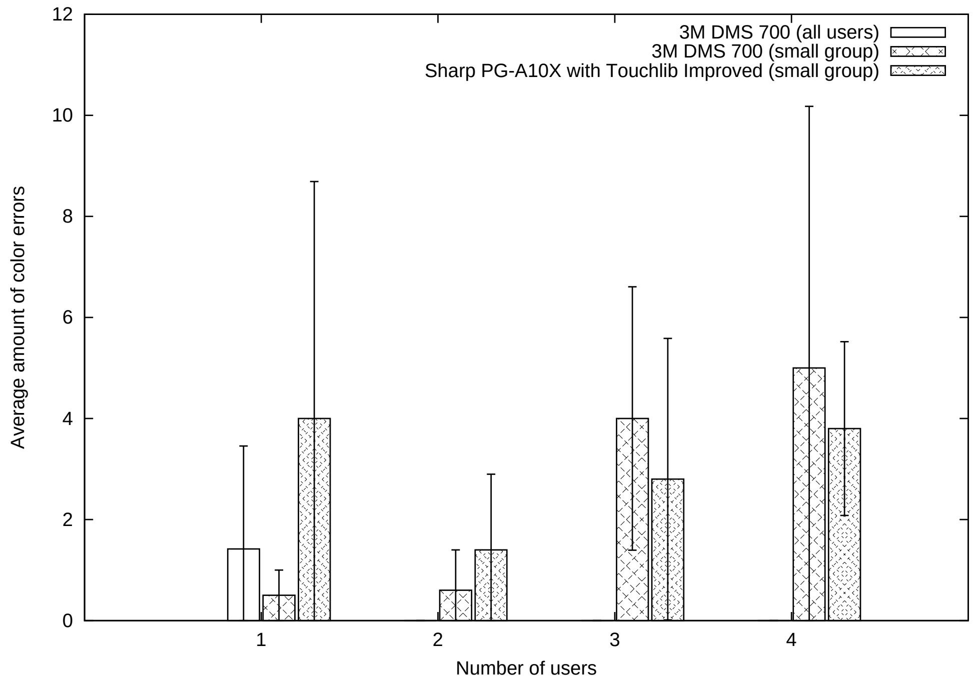


Figure 5.25: Average number of color mistakes when completing the selection task.

CHAPTER 6

Discussion

We evaluated the multi-touch device through a set of experiments. In this chapter we discuss the results based on the hypotheses we set in the previous chapter.

6.1 The performance of a task on a multi-touch device depends on the used hardware

From the experiment results it becomes evident that system performance has a high impact. Due to the latency of our multi-touch system using the 3M projector (and the Touchlib 'bug'), results show longer task completion times (compared to the Sharp projector). Based on the questionnaire (see Appendix C) most users (40%) rated the responsiveness of the system as fair.

Limited to the used camera and table size, our multi-touch table has a precision of $\frac{947mm}{640pixel} = 1.48mm/pixel$. The influence of the precision becomes visible when users are required to dock objects within a tolerance. While pointing objects is no issue on a multi-touch table, the time spent to correct the position of an object is. As a result of the system latency, movement is not directly visible. Therefore users are required to anticipate on the result of movement. In these case the performance of completing a task is reduced because of users waiting to view the result. Another problem related to the precision is the sensitivity of the multi-touch table to touch. When using RI, fingertips are tracked based on contrast differences and contours. However, when a finger is near the surface but not on the surface, the fingertip still reflects IR light to the camera. While this problem can be solved partially by adjusting the image threshold in Touchlib, it will also reduce the sensitivity to touch (in this case a touch requires more pressure to increase the size of the contact and reflect enough infrared light).

Results of the questionnaire show that the ergonomics of the table should be improved. Due to the design of the table, it is not possible to place your feet under the table. This increases the distance with the touch sensitive area of the multi-touch screen. Another problem is the size of the table. Tall users complain about having back stains as result of hanging over the table while short users are having trouble to reach objects at the top of the screen. The most important problem users mentioned is the high friction with the multi-touch surface. The surface which is made of acrylic makes it difficult to perform smooth motions over long distances. After completing all experiments users mentioned painful fingertips.

6.2 The comparison between input devices on task performance

When comparing the performance of a task in one dimension with the mouse device, the break even point of Test A is positioned at an ID of 0.8 bits (when using the Sharp projector), and an ID of 2.72 bits (when using the 3M projector). From our data set, the closes entry to the ID of 2.72 bits is the ID of 2.61 bits. This corresponds to a task with an object width of 50 pixels (4.62 cm) and an amplitude of 256 pixels (23.68 cm).

In the results of the IP of the same test, the IP of the multi-touch device (using the 3M projector) contains a very high bandwidth value. The reason for this behavior is related to the design of the task. Originally the Fitts' Law test was designed to measure the performance of single-input user devices. On a desktop, an indirect device (the mouse) is used to move a cursor to its destination. In order to reach each destination, it is required to move the cursor from its original position to a new position. On the multi-touch table no cursor was present. Users were allowed to use both hands in order to select the objects. Because no cursor is present, users often used the hand which was the closest to the object. Depending on the positioning of the hand, the required travel distance was minimal resulting in low touch times.

Results of Test B of the first experiment show different behavior compared to Test A. By changing the experiment to a two dimensional one, the break even point is positioned at an ID of 1.6 bits (when using the Sharp projector) and 2.48 bits (when using the 3M projector). A task of 1.6 bits corresponds with an object width of 30 pixels (2.77 cm) and an amplitude of 64 pixels (5.92 cm).

Results of the IP of Test B show that the mouse device outperforms the multi-touch device using the 3M projector. Only the multi-touch device using the Sharp projector is capable of completing these tasks faster. The reason for the lower performance can be found in the way users interact with the devices. On the desktop, it does not require much effort to select objects which are placed near the top of the screen. On a multi-touch table, it requires more physical effort to select small objects near the top of the screen. From the results of the questionnaire, 75% of the test users preferred to complete this set of tasks on the multi-touch device.

In tasks which contain object manipulation, gesture based interaction helps to reduce the time to complete this task. Complex tasks (Figure 5.12) containing objects in a different rotation and scaling requires multiple actions (Figure 5.14) with the mouse device to complete. When performing the same set of (difficult) tasks the total completion time and the number of actions is reduced in most tasks (Using the multi-touch device with the Sharp projector). Results from the questionnaire show that 50% of the users preferred to complete this set of task on the desktop and the other 50% the multi-touch device.

Based on these results, we can confirm that tasks which require precision are faster performed with a mouse device and complex mouse tasks can be performed faster when using the the multi-touch device.

6.3 The impact of collaboration on a multi-touch device on task performance

Results from experiment 3 and 4 confirm our hypotheses. Experiment three showed that when more users are sorting collaboratively the task completion time is reduced. When using the system with the Sharp projector, four users were able to reduce the tasks time to 27% of the original time. Although the task completion time decrease, users tend to be less precise. According to the results, users are more likely to make sorting mistakes. Collaboration on a multi-touch also points out that working on a small surface can be troublesome. From observation we noticed that users tend to sort objects that are in reaching distance first. When a user moves a tile to the container, the area around the container is blocked by the hand for a short period of time causing other users to stall. This problem increases, when more users are active or when users are moving multiple tiles at the same time.

In the pointing task of experiment 4, the task completion time is reduced when the number of users is increased. From observation we noticed that when four users are completing the task, it often occurs that users are trying to reach for the same object. As a results the number of selection errors is increased. In the selection task a similar effects compared to the pointing task is visible. Increasing the number of users reduces the task completion time. From the questionnaire 40% of the users rated the task difficulty as rather difficult. Selecting objects on the surface was difficult due to the high friction of the acrylic. Another problem was the sensitivity of touch at the corners of the multi-touch screen that required more pressure to take effect.

CHAPTER 7

Conclusion and Future work

With the construction of our own camera based multi-touch table we have demonstrated that multi-touch technology has become affordable. The precision, reliability and responsiveness of a multi-touch table depends on the used hardware. Because of the use of a camera based multi-touch technology, ambient light has a large influence in the tracking performance.

In our set of demo applications we have demonstrated the power of multi-touch devices. Compared to desktop applications, users were able to manipulate objects in a natural way by touch. Applications such as the real-time fluid dynamics simulations demonstrated that scientific applications can benefit from multi-user input. NASA WorldWind is a demonstration of how existing applications can benefit from multi-touch input using gesture based interaction.

We have evaluated the performance of our multi-touch table through a set of experiments. Results show that our current hardware using the a high latency projector influenced our test results. While multi-touch is capable of performing some task faster than the mouse device, it should not be considered as a replacement. When tasks require precision, the multi-touch device shows longer task completion times with higher error rates. Multi-touch device however, encourage collaboration. Our test results show significant improvement when the number of users is increased.

Future work

Although we have presented a complete multi-touch solution, there is still room for improvement on hardware and software level. A few suggestions:

- Latency results show a large performance hit when correcting the camera image from barrel distortion. Instead of correcting the entire camera image, it is also possible to apply position correction on the output of the blob tracker.
- Currently image processing is done on the CPU. In order to reduce the load on the CPU it is possible to use graphics processing units (GPU) for image processing. A port of Intel's OpenCV library that uses the GPU has recently been made available (GPUCV [10])
- The current blob tracker used in Touchlib is not scalable. Depending on the used hardware, Touchlib will fail to track fifty or more blobs in real-time. A smarter algorithm would not only take the distance into account but also the region and direction of a blob. It would also be interesting to use the GPU for blob tracking.
- Based on our own experiences and feedback of our test users, the touch surface should be improved. The friction with the acrylic makes it hard to use the system for a longer period of time. Tests with hardened glass showed promising results.
- Currently alphanumeric input on a virtual keyboard is a difficult task due the lack of tactile feedback and the low responsiveness of the system. The responsiveness can be improved by using a camera that captures more than 30 frames per second.

APPENDIX A

Diffuser materials

Based on published papers on camera based multi-touch devices, most of the used materials are known. However most papers do not describe the type or brand of material used for the diffuser. Therefore it was required to try out our own materials. Based on reports [38] that tracing paper would work we tried it out on our FTIR prototype with success. However when using the same material on the table which used RI the diffuser material proved not to perform well enough. Normal tracing paper absorbs too much infrared light causing glare. Early prototypes of MS Surface (which is based on RI) stated to use 'architect vellum' as a diffuser. Unfortunately the material was not available at the local paper shops. Therefore we needed to find an alternative. By comparing the candidate material's on opacity, strength and thickness a selection of five paper-like materials was made. Each of these materials was tested with FTIR and RI under the same testing conditions.

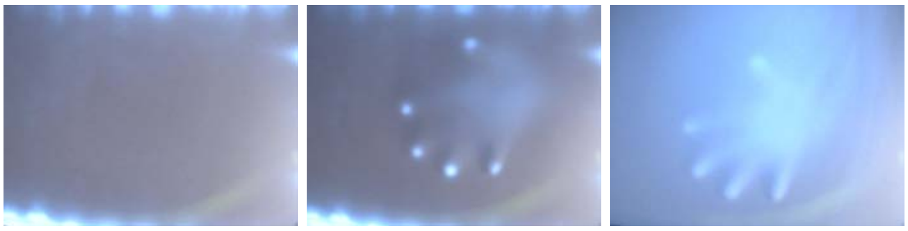
A.1 Tested diffuser materials

- Calqueer 90 g/m^3
- Cromatico extra white
- Polyester film 2x matte
- Tracing paper
- Translucent pearl

A.2 Testing conditions

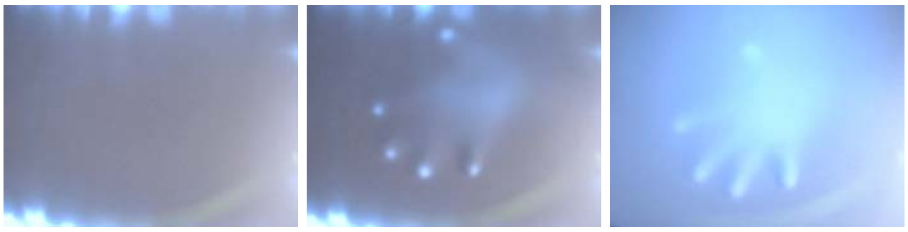
- All measurements were done on a display size of 40×30 cm.
- All diffuser materials have the dimensions of A3 (297×420 mm).
- When using RI one infrared illuminator was used containing 20 infrared LEDs (Osram SFH485P)
- The distance between the infrared illuminator and the screen was 40 cm.
- The infrared illuminator was placed under the camera.
- To prevent glare the illuminator was placed under an angle of 45 degrees.

A.3 Materials comparison results



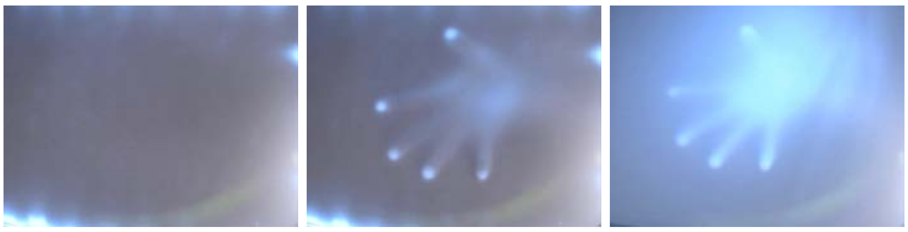
(a) FTIR empty (b) FTIR touched (c) RI touched

Figure A.1: Camera test results using Calqueer 90 g/m^3.



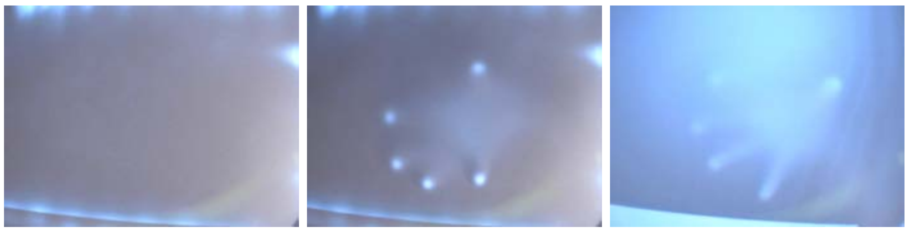
(a) FTIR empty (b) FTIR touched (c) RI touched

Figure A.2: Camera test results using Cromatico extra white.



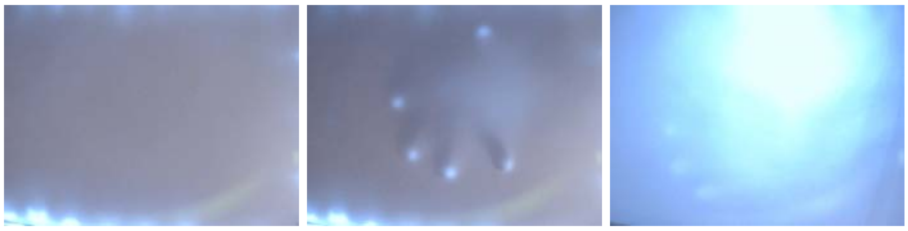
(a) FTIR empty (b) FTIR touched (c) RI touched

Figure A.3: Camera test results using Polyester film 2x matte.



(a) FTIR empty (b) FTIR touched (c) RI touched

Figure A.4: Camera test results using Tracing paper.



(a) FTIR empty (b) FTIR touched (c) RI touched

Figure A.5: Camera test results using Translucent pearl.

APPENDIX B

Touchlib Reference

Currently no official Touchlib documentation is available. Only a few configuration examples are available in the SVN tree. This appendix provides general information about configuring and using Touchlib.

B.1 Project details

Project site: www.touchlib.com
SVN: http://code.google.com/p/touchlib/
SVN (repository): http://touchlib.googlecode.com/svn/multitouch/

B.2 Touchlib config.xml

The configuration file consist out of the following parts:

1. XML version definition
   example: <?xml version="1.0" ?>
2. Tracker configuration
   example:
   <blobconfig distanceThreshold="250" minDimension="2" maxDimension="250" ghostFrames="0" minDisplacementThreshold="2.000000" />
   Tolerance settings of the blobtracker. The distanceThreshold contains the value of how many pixels a blob can travel. The minDimension and maxDimension variables specify how small or large a contour can be to be detected as a touch. The value of ghostFrames specifies the number of extra frames Touchlib should use for the blobtracker. The minDisplacementThreshold specifies how many pixels a contour needs to be moved before calling the update event.
3. Bounding box
   example:
   <bbox ulX="0.000000" ulY="0.000000" lrX="1.000000" lrY="1.000000" />
   Specifies in which region blob detection should be applied.
4. Screen calibration points
   example: <screen>...[points]...</screen>
   These values will be filled when running the configapp.
5. The filtergraph
   example: <filtergraph>...[filters]...</filtergraph>
   Filters are explained in the next section.

B.3 Touchlib filters

General note: All filters need to be in between of the filtergraph tags.

$$< filtergraph > ... < /filtergraph >$$

The first filter in the filtergraph needs to be a video capture filter, the last one must be the rectify filter.

B.3.1 Video capture filters

cvcapture

Description:
This is the default video capture filter used by Touchlib. It uses OpenCV function to capture a video stream. It is also possible to use a video file for testing purposes.
Usage (Video file):
```
<cvcapture label="cvcapture">
<source value="../tests/rear4.avi" />
</cvcapture>
```

Usage (Camera):
```
<cvcapture label="cvcapture">
<source value="cam" />
</cvcapture>
```

dsvlcapture

Description:
This is an implementation which uses DirectShow to capture a video stream.
Usage:
```
<dsvlcapture label="dsvlcapture" />
```

cmucapture

Description:
This filter uses the CMU driver to access FireWire video capture devices.
Usage:
```
<cmucapture label="cmucapture">
<brightness value="-1" />
<exposure value="-1" />
<flipRGB value="false" />
<gain value="87" />
<gamma value="1" />
<mode value="640x480mono" />
<rate value="30fps" />
<saturation value="90" />
<sharpness value="80" />
<whitebalanceH value="0" />
<whitebalanceL value="-1" />
</cmucapture>
```

vwcapture

Description:
This filter uses the VideoWrapper API to access FireWire video capture devices.
Usage:
```
<vwcapture label="capture1">
<videostring value="pgr: 0 640 30 grey16 1 rgb" />
</vwcapture>
```

The videostring contain the camera parameters which depends on the interface type:
Cameras using DCAM specifications from Videre Design
Parameters: "dcam: camNum width frameRate colorMode scale"
Example: "dcam: 0 640 30 rgb 2"

Cameras from Point Grey Research
Parameters: "pgr: camNum width frameRate colorMode scale outputMode"
Example: "pgr: 0 640 30 grey8 1 rgb"

Cameras using VidCapture (DirectShow)

Parameters: ”vc: camNum width frameRate colorMode scale outputMode”
Example: ”vc: 0 640 15 rgb 0”

B.3.2 Video processing filters

Mono filter

Description:
Touchlib requires an 8 bit grey scale source image. This filter is only required if the video capture filter is not capable of delivering the right image format.
Usage:
<mono label=”monofilter” />

Background Filter

Description:
This filter removes the background by creating a reference image on initialization and subtracting it from the current active frame.
Usage:
<backgroundremove label=”backgroundfilter”>
<threshold value=”20” />
</backgroundremove>
The threshold value has a range from 0-255.

Smoothing Filter

Description:
The smoothing filter applies a gaussian blur on the source image.
Usage:
<smooth label=”smoothfilter” />

Invert Filter

Description:
The filter inverts a greyscale image. This filter is only required for FI.
Usage:
<invert label=”invert” />

Scaler Filter

Description:
If the previous used filter gives a weak output the scaler filter is used to amplify the current image.
Usage:
<scaler label=”scaler” >
<level value=”70” />
</scaler>

Brightness and Contrast Filter

Description:
In particular situation it might be required to adjust the image brightness and contrast. This filter is only used for FTIR
Usage:
<brightnesscontrast label=”brightnesscontrast4”>
<brightness value=”0.1” />
<contrast value=”0.4” />
</brightnesscontrast>

Highpass Filter

Description:
When using RI or FI, the brightness of the blobs is much weaker compared to FTIR. However if enough contrast is available the HighpassFilter is capable of amplifying the output of these blobs.
Usage:

```
<highpass label=”highpass”>
<filter value=”6” />
<scale value=”32” />
</highpass>
```

Highpass Filter (Simple)

Description:
Same purpose as the previous filter, however it uses a simpler algorithm and therefore performance faster than the default HighpassFilter. The filter contains two different methods to amplify the source images, currently it is recommended to set the *noiseMethod* to 1.
Usage:

```
<simplehighpass label=”simplehighpass”>
<blur value=”13” />
<noise value=”3” />
<noiseMethod value=”1” />
</simplehighpass>
```

Barrel Distortion Correction Filter

Description:
A wide-angle lens often contains a radial distortion which can not be corrected by Touchlib default correcting algorithm. This filter corrects the barrel distortion based on the lens characteristics. The filter requires a camera.yml file which is create by the barrel distortion correction tool. Because the image get corrected, the result might be missing out important parts. By setting the border size this can be corrected (if not needed, set this to 0).

```
<barreldistortioncorrection label=”barreldistortioncorrection1”>
<border_size value=”20” />
</barreldistortioncorrection>
```

Crop Filter

Description:
Allows the user crop the video image.
The values posX and posY define the top left position of the crop area. The height and width define the crop area size.

```
<crop label=”crop”>
<posX value=”40” />
<posY value=”40” />
<height value=”120” />
<width value=”160” />
</crop>
```

Rectify Filter

Description:
This is the final filter of the image processing pipeline. The threshold is set to a value in which blobs are visible but noise input is ignored. This image will be used to perform blob detection on.
Usage:

```
<rectify label=”rectify”>
<level value=”75” />
```

</rectify>
The level value has a range from 0-255.

B.4 Touchlib calibration

In order to calibrate the Touchlib it is required to have a fully functioning multi-touch table (this includes a camera and projector). It is recommended to look at the example xml files to create a config.xml configuration file. Every multi-touch technique requires a different filter chain. When the filter chain has been set up, the configuration application can be used by starting configapp.exe from the Touchlib bin directory (or ./configapp from the /src directory under linux). It now displays the filters you have entered in the config.xml file. Adjust the sliders in the filters until the rectify filter only shows clear blobs when being touched. If you need to recapture the background press 'b'.

When you are satisfied with the result of the filter chain you can start the actual calibration. Press 'enter' to go into full screen mode. The config application shows a black background with 20 green colored reference points, 5 horizontal and 4 vertical. In the upper left corner you will see an example of the camera input. If you do not want to use the full size of your camera input it is possible to adjust the bounding box by pressing 'x'. Please read the instruction on the screen to adjust this box. To start the calibration press 'c'. The current point to select is red, including green arrows rotating around the spot. If your camera is orientated differently (such as flipped on the Y-axis) Touchlib will auto correct this on the first touch. Continue touching the required points. When the calibration is finished you can test the accuracy by touching the surface. It shows a rectangle shaped to the size of the 'press'. To exit the configuration press 'ESC'. When the application quits all values are written to the config.xml. If you need to adjust the calibration, you can adjust the values inside the <screen>...</screen> tag.

B.5 TouchData structure

The range of the X and Y are from 0 to 1 (floating point).

Data type	**Variable name**	**Properties**
integer	ID	Unique identifier blob
integer	tagID	Unique identifier fiducial
float	X	Horizontal position
float	Y	Vertical position
float	height	Hight of fiducial
float	width	Width of fiducial
float	angle	Rotation of fiducial
float	area	Size of blob (pressure)
float	dX	Delta movement horizontal axis
float	dY	Delta movement vertical axis

Table B.1: TouchData structure

APPENDIX C

Questionnaire

Experiment 1: Repetitive object pointing task

1. How well did the mouse device perform for this task?
   a) Very poor (0%)
   b) Poor (5%)
   c) Fair (5%)
   d) Good (90%)
   e) Very good (0%)
2. How well did the multi-touch device perform for this task?
   a) Very poor (0%)
   b) Poor (0%)
   c) Fair (20%)
   d) Good (35%)
   e) Very good (45%)
3. Which input device did you prefer for this set of tasks?
   a) Mouse (20%)
   b) Multi-touch device (75%)
   c) No preference (5%)

Experiment 2: Object manipulation

4. How well did the mouse device perform for this task?
   a) Very poor (0%)
   b) Poor (5%)
   c) Fair (25%)
   d) Good (60%)
   e) Very good (10%)
5. The mouse sensitivity for movement was:
   a) Too little (10%)
   b) Just right (90%)
   c) Too much (0%)
6. The mouse sensitivity for rotation was:
   a) Too little (15%)
   b) Just right (75%)

   c) Too much (10%)

7. The mouse sensitivity for scaling was:
   a) Too little (5%)
   b) Just right (85%)
   c) Too much (10%)

8. How well did the multi-touch device perform for this task?
   a) Very poor (0%)
   b) Poor (5%)
   c) Fair (60%)
   d) Good (30%)
   e) Very good (5%)

9. The multi-touch sensitivity for movement was:
   a) Too little (40%)
   b) Just right (50%)
   c) Too much (10%)

10. The multi-touch sensitivity for rotation was:
    a) Too little (25%)
    b) Just right (60%)
    c) Too much (15%)

11. The multi-touch sensitivity for scaling was:
    a) Too little (10%)
    b) Just right (80%)
    c) Too much (10%)

12. How easy was it for you to perform a gesture to manipulate objects (multi-touch only)?
    a) Very difficult (0%)
    b) Rather difficult (15%)
    c) Fair (40%)
    d) Moderately easy (45%)
    e) Very easy (0%)

13. Which input device do you prefer for this set of tasks?
    a) Mouse (50%)
    b) Multi-touch device (50%)

Experiment 3: Collaborative sorting task

14. How easy was it for you to perform a parallel dragging task?
    a) Very difficult (0%)
    b) Rather difficult (25%)
    c) Fair (25%)
    d) Moderately easy (35%)
    e) Very easy (15%)

Experiment 4: Collaborative point and selection task

Experiment 4a (Pointing task)

15. How easy was it for you to perform a parallel pointing task?
    a) Very difficult (0%)
    b) Rather difficult (10%)
    c) Fair (15%)
    d) Moderately easy (25%)
    e) Very easy (50%)

Experiment 4b (Selection task)

16. How easy was it for you to perform a parallel selecting task (lasso)?
    a) Very difficult (5%)
    b) Rather difficult (40%)
    c) Fair (10%)
    d) Moderately easy (25%)
    e) Very easy (20%)

Hardware

17. How well did the multi-touch device perform in responsiveness?
    a) Very poor (0%)
    b) Poor (20%)
    c) Fair (40%)
    d) Good (35%)
    e) Very good (5%)
18. How well did the multi-touch device perform in precision?
    a) Very poor (0%)
    b) Poor (30%)
    c) Fair (45%)
    d) Good (20%)
    e) Very good (5%)

Ergonomics

19. Did you feel any of the following problems when using the desktop (mouse) (you may select multiple options)
    a) General discomfort
    b) Problems focusing
    c) Watery eyes
    d) Headache
    e) Other: ...
20. Did you feel any of the following problems when using the multi-touch device (you may select multiple options)
    a) General discomfort
    b) Problems focusing
    c) Watery eyes
    d) Headache
    e) Other: ...

General questions

21. What level of experience do you have with multi-touch devices:?
    a) Beginner (70%)
    b) Novice (15%)
    c) Intermediate (15%)
    d) Advanced (0%)
22. Did you feel you required more training to use an multi-touch device?
    a) Yes (45%)
    b) No (55%)
23. Which task did you find the most difficult to perform?
    a) Experiment 1 (clicking objects) (0%)
    b) Experiment 2 (object manipulation) (80%)
    c) Experiment 3 (sorting task) (5%)
    d) Experiment 4a (clicking objects in the right order) (0%)
    e) Experiment 4b (selecting objects in the right order) (15%)
24. Were there special situations or personal (dis)abilities that influenced your participation?
    a) No (90%)
    b) Yes, namely: ... (10%)
25. How did you experienced multi-touch interaction in general?
    a) Very poor (0%)
    b) Poor (0%)
    c) Fair (20%)
    d) Good (55%)
    e) Very good (25%)

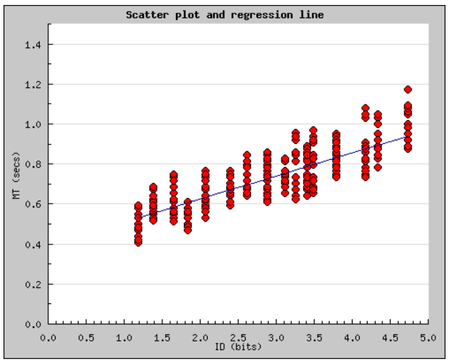
Feedback

26. Any remarks or personal opinions about the tests?

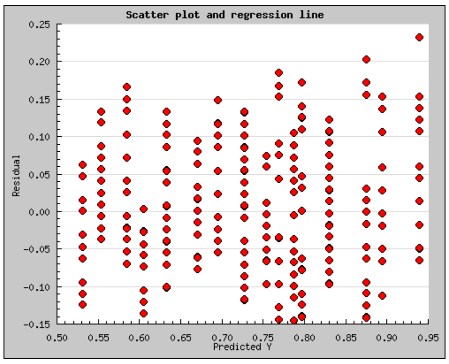
APPENDIX D

Results Fitts test

D.1 Results 1D test

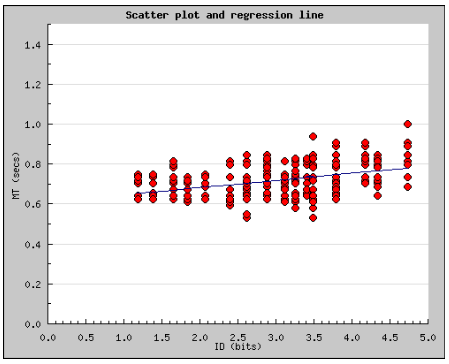


(a) Regression line

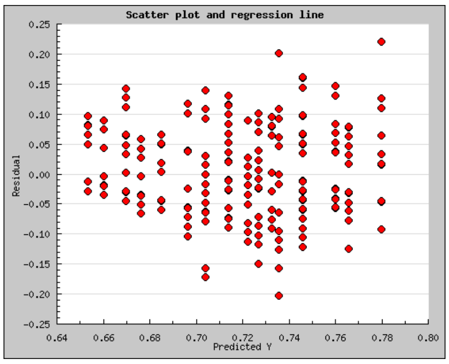


(b) Residual

Figure D.1: The left image shows a scatter plot of the mouse device data set. The x-axis represents the index of difficulty (ID), the y-axis the measured time (MT) to complete the task. Linear regression is performed on the data set. The right image shows the residual, which compares the predicted measurement time with the actual measurement time.

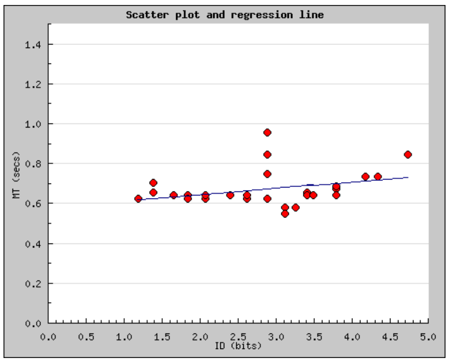


(a) Regression line

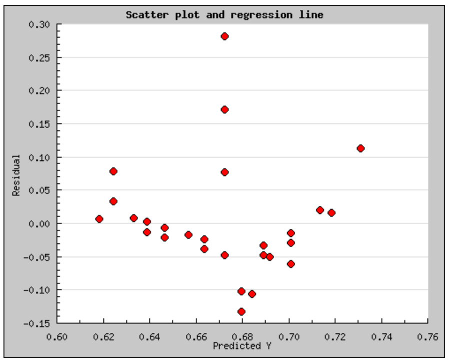


(b) Residual

Figure D.2: The left image shows a scatter plot of the multi-touch data set using the 3M digital projector. The x-axis represents the index of difficulty (ID), the y-axis the measured time (MT) to complete the task. Linear regression is performed on the data set. The right image shows the residual, which compares the predicted measurement time with the actual measurement time.

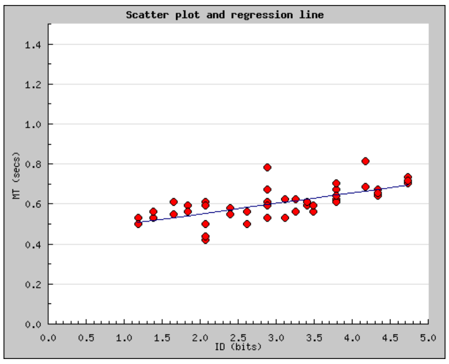


(a) Regression line

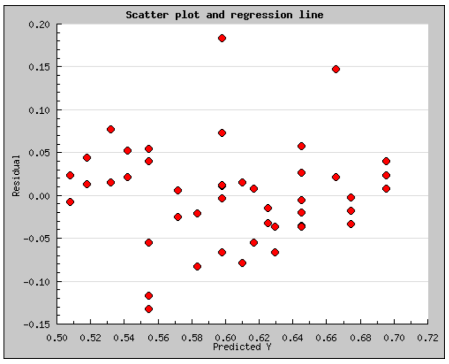


(b) Residual

Figure D.3: The left image shows a scatter plot of the multi-touch data set (small group) using the 3M digital projector. The x-axis represents the index of difficulty (ID), the y-axis the measured time (MT) to complete the task. Linear regression is performed on the data set. The right image shows the residual, which compares the predicted measurement time with the actual measurement time.



(a) Regression line

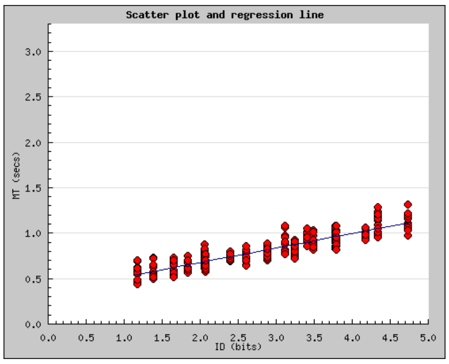


(b) Residual

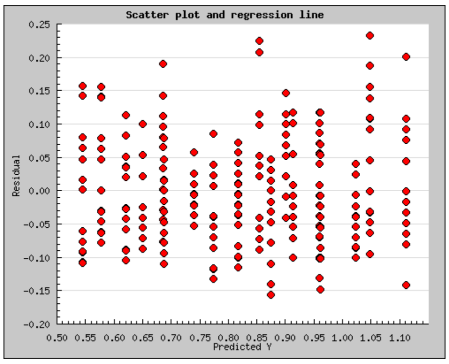
Figure D.4: The left image shows a scatter plot of the multi-touch data set (small group) using the Sharp digital projector. The x-axis represents the index of difficulty (ID), the y-axis the measured time (MT) to complete the task. Linear regression is performed on the data set. The right image shows the residual, which compares the predicted measurement time with the actual measurement time.

.

D.2 Results 2D test

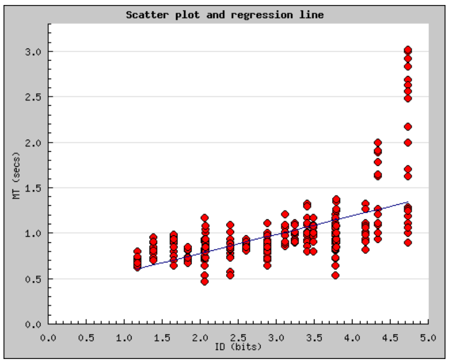


(a) Regression line

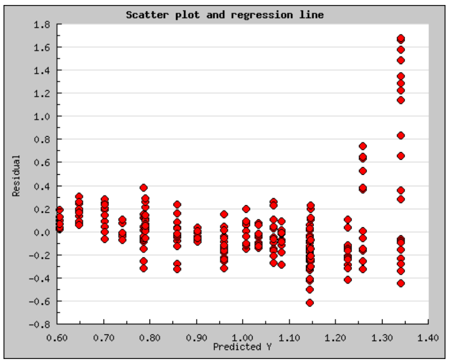


(b) Residual

Figure D.5: The left image shows a scatter plot of the mouse device data set. The x-axis represents the index of difficulty (ID), the y-axis the measured time (MT) to complete the task. Linear regression is performed on the data set. The right image shows the residual, which compares the predicted measurement time with the actual measurement time.

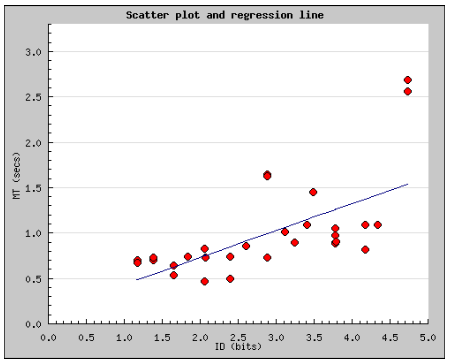


(a) Regression line

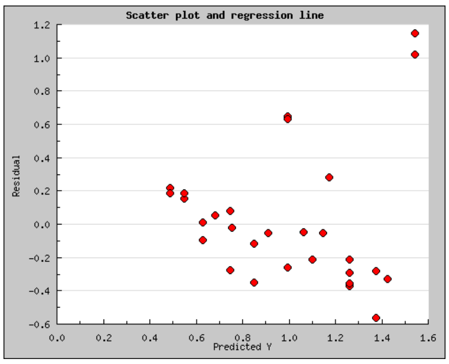


(b) Residual

Figure D.6: The left image shows a scatter plot of the multi-touch data set using the 3M digital projector. The x-axis represents the index of difficulty (ID), the y-axis the measured time (MT) to complete the task. Linear regression is performed on the data set. The right image shows the residual, which compares the predicted measurement time with the actual measurement time.

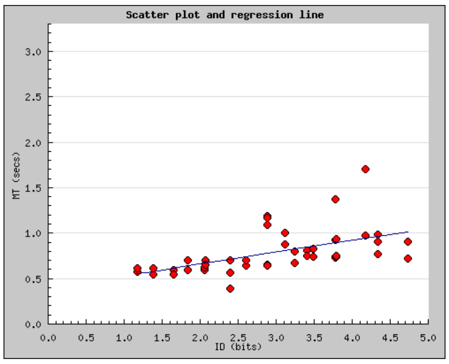


(a) Regression line

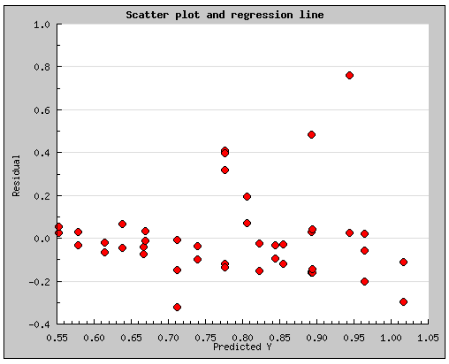


(b) Residual

Figure D.7: The left image shows a scatter plot of the multi-touch data set (small group) using the 3M digital projector. The x-axis represents the index of difficulty (ID), the y-axis the measured time (MT) to complete the task. Linear regression is performed on the data set. The right image shows the residual, which compares the predicted measurement time with the actual measurement time.



(a) Regression line



(b) Residual

Figure D.8: The left image shows a scatter plot of the multi-touch data set (small group) using the Sharp digital projector. The x-axis represents the index of difficulty (ID), the y-axis the measured time (MT) to complete the task. Linear regression is performed on the data set. The right image shows the residual, which compares the predicted measurement time with the actual measurement time.

Bibliography

[1] Ross Bencina and Martin Kaltenbrunner. The design and evolution of fiducials for the reactivision system. In *Proceedings of the 3rd International Conference on Generative Systems in the Electronic Arts (3rd Iteration 2005)*, Melbourne, Australia, 2005.

[2] Didier Brun. Mouse gesture recognition. 2007. URL `http://www.bytearray.org/?p=91`.

[3] Mat Buckland and Mark Collins. *AI Techniques for Game Programming*. Premier Press, 2002. ISBN 193184108X.

[4] W. Buxton and B. Myers. A study in two-handed input. In *CHI '86: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 321–326, New York, NY, USA, 1986. ACM Press. ISBN 0-89791-180-6. doi: http://doi.acm.org/10.1145/22627.22390.

[5] William Buxton, Ralph Hill, and Peter Rowley. Issues and techniques in touch-sensitive tablet input. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 215–224, New York, NY, USA, 1985. ACM Press. ISBN 0-89791-166-0. doi: http://doi.acm.org/10.1145/325334.325239.

[6] NASA Ames Research Center. Nasa world wind. 2004. URL `http://worldwind.arc.nasa.gov/`.

[7] Microsoft Corporation. MS surface. 2007. URL `http://www.microsoft.com/surface/`.

[8] Paul Dietz and Darren Leigh. Diamondtouch: a multi-user touch technology. In *UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 219–226, New York, NY, USA, 2001. ACM Press. ISBN 1-58113-438-X. doi: http://doi.acm.org/10.1145/502348.502389.

[9] Ralph G. Johnson et al. US patent #3,673,327: Touch actuable data input panel assembly, 1972. URL `http://www.google.com/patents?vid=USPAT3673327`.

[10] Jean-Philippe Farrugia and Patrick Horain. GPUCV: A framework for image processing acceleration with graphics processors. In *International Conference on Multimedia and Expo (ICME)*, July 2006. URL `http://liris.cnrs.fr/publis/?id=2469`. On the CDROM proc.of the IEEE International Conference on Multimedia & Expo (ICME 2006).

[11] Paul M. Fitts. The information capacity of the human motor system in controlling the amplitude of movement. In *Journal of Experimental Psychology*, pages 381–391, 1954.

[12] Clifton Forlines, Daniel Wigdor, Chia Shen, and Ravin Balakrishnan. Direct-touch vs. mouse input for tabletop displays. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 647–656, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-593-9. doi: http://doi.acm.org/10.1145/1240624.1240726.

[13] Jefferson Y. Han. Low-cost multi-touch sensing through frustrated total internal reflection. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 115–118, New York, NY, USA, 2005. ACM Press. ISBN 1-59593-271-2. doi: http://doi.acm.org/10.1145/1095034.1095054.

[14] Jefferson Y. Han. Multi-touch interaction wall. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Emerging technologies*, page 25, New York, NY, USA, 2006. ACM Press. ISBN 1-59593-364-6. doi: http://doi.acm.org/10.1145/1179133.1179159.

[15] Peter Hutterer. Multi-pointer x server. 2007. URL `http://wearables.unisa.edu.au/mpx/`.

[16] Intel. Open source computer vision library: Reference manual. 2001. URL `http://www.intel.com/technology/computing/opencv/`.

[17] Shahram Izadi, Steve Hodges, Alex Butler, Alban Rrustemi, and Bill Buxton. Thinsight: integrated optical multi-touch sensing through thin form-factor displays. In *EDT '07: Proceedings of the 2007 workshop on Emerging displays technologies*, page 6, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-669-1. doi: http://doi.acm.org/10.1145/1278240.1278246.

[18] JazzMutant. Lemur input device. 2006. URL `http://www.jazzmutant.com/lemur_overview.php`.

[19] Sergi Jordà, Martin Kaltenbrunner, Günter Geiger, and Ross Bencina. The reacTable*. In *Proceedings of the International Computer Music Conference (ICMC 2005)*, Barcelona, Spain, 2005.

[20] Leonard R. Kasday. US patent #4,484,179: Touch position sensitive surface, 1984. URL `http://www.google.com/patents?vid=USPAT4484179`.

[21] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. Technical Report 8, 1966.

[22] I. Scott MacKenzie. Movement time prediction in human-computer interfaces. In *Proceedings of the conference on Graphics interface '92*, pages 140–150, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc. ISBN 0-9695338-1-0.

[23] James B. Mallos. US patent #4,346,376: Touch position sensitive surface, 1982. URL `http://www.google.com/patents?vid=USPAT4346376`.

[24] Nobuyuki Matsushita and Jun Rekimoto. Holowall: designing a finger, hand, body, and object sensitive wall. In *UIST '97: Proceedings of the 10th annual ACM symposium on User interface software and technology*, pages 209–210, New York, NY, USA, 1997. ACM Press. ISBN 0-89791-881-9. doi: http://doi.acm.org/10.1145/263407.263549.

[25] Robert E. Mueller. US patent #3,846,826: Direct television drawing and image manipulation system, 1974. URL `http://www.google.com/patents?vid=USPAT3846826`.

[26] Laurence Muller. Building a multi-touchscreen (based on FTIR). 2007. URL `http://www.multigesture.net/articles/`.

[27] Kenji Oka, Yoichi Sato, and Hideki Koike. Real-time fingertip tracking and gesture recognition. *IEEE Comput. Graph. Appl.*, 22(6):64–71, 2002. ISSN 0272-1716. doi: http://dx.doi.org/10.1109/MCG.2002.1046630.

[28] Thomas Pintaric. Directshow video processing library. 2005. URL `http://sourceforge.net/projects/dsvideolib`.

[29] Jun Rekimoto. Smartskin: an infrastructure for freehand manipulation on interactive surfaces. In *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 113–120, New York, NY, USA, 2002. ACM Press. ISBN 1-58113-453-3. doi: http://doi.acm.org/10.1145/503376.503397.

[30] Stephan Rusdorf, Guido Brunnett, Mario Lorenz, and Tobias Winkler. Real-time interaction with a humanoid avatar in an immersive table tennis simulation. *IEEE Transactions on Visualization and Computer Graphics*, 13(1):15–25, 2007. ISSN 1077-2626. doi: http://doi.ieeecomputersociety.org/10.1109/TVCG.2007.18.

[31] Martin Kaltenbrunner Sergi Jord, Marcos Alonso and Gnter Geiger. reacTable: an electro-acoustic music instrument with a tabletop tangible user interface. 2007. URL `http://reactable.iua.upf.edu/`.

[32] J. Stam. Real-time fluid dynamics for games, 2003. URL `citeseer.ist.psu.edu/stam03realtime.html`.

[33] Jos Stam. A simple fluid solver based on the FFT. *Journal of Graphics Tools: JGT*, 6(2):43–52, 2001. URL `citeseer.ist.psu.edu/stam02simple.html`.

[34] Lucia Terrenghi, David Kirk, Abigail Sellen, and Shahram Izadi. Affordances for manipulation of physical versus digital media on interactive surfaces. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1157–1166, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-593-9. doi: http://doi.acm.org/10.1145/1240624.1240799.

[35] Unibrain. Fire-i digital board camera. 2008. URL `http://www.unibrain.com/Products/VisionImg/Fire_i_BC.htm`.

[36] Carnegie Mellon University. Cmu 1394 digital camera driver. 2008. URL `http://www.cs.cmu.edu/~iwan/1394/`.

[37] Harry van der Veen. FTIR multi-touch display how-to guide (alpha version 0.2). 2007. URL `http://www.multitouch.nl/documents/multitouchdisplay_howto_070523_v02.pdf`.

[38] David Wallin. Touchlib: an opensource multi-touch framework. 2006. URL `http://www.whitenoiseaudio.com/touchlib/`.

[39] Eric W. Weisstein. Barycentric coordinates. 2008. URL `http://mathworld.wolfram.com/BarycentricCoordinates.html`.

[40] Eric W. Weisstein. Areal coordinates. 2008. URL `http://mathworld.wolfram.com/ArealCoordinates.html`.

[41] Richard M. White. US patent #4,484,179: Tactile sensor employing a light conducting element and a resiliently deformable sheet, 1987. URL `http://www.google.com/patents?vid=USPAT4484179`.

[42] Wikipedia. Total internal reflection. 2008. URL `http://en.wikipedia.org/wiki/Total_internal_reflection#Frustrated_total_internal_reflection`.

[43] Yves Guiard Abigail Sellen William Buxton, Mark Billinghurst and Shumin Zhai. Human input to computer systems: Theories, techniques and technology. 1994. URL `http://www.billbuxton.com/inputManuscript.html`.

[44] Andrew D. Wilson. Touchlight: an imaging touch screen and display for gesture-based interaction. In *ICMI '04: Proceedings of the 6th international conference on Multimodal interfaces*, pages 69–76, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-995-0. doi: http://doi.acm.org/10.1145/1027933.1027946.

[45] Andrew D. Wilson. Playanywhere: a compact interactive tabletop projection-vision system. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 83–92, New York, NY, USA, 2005. ACM Press. ISBN 1-59593-271-2. doi: http://doi.acm.org/10.1145/1095034.1095047.

[46] Carl D. Worth. xstroke: Full-screen gesture recognition for x. 2003. URL `http://www.usenix.org/events/usenix03/tech/freenix03/full_papers/worth/worth_html/xstroke.html`.

[47] Mike Wu and Ravin Balakrishnan. Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. In *UIST '03: Proceedings of the 16th annual ACM symposium on User interface software and technology*, pages 193–202, New York, NY, USA, 2003. ACM Press. ISBN 1-58113-636-6. doi: http://doi.acm.org/10.1145/964696.964718.