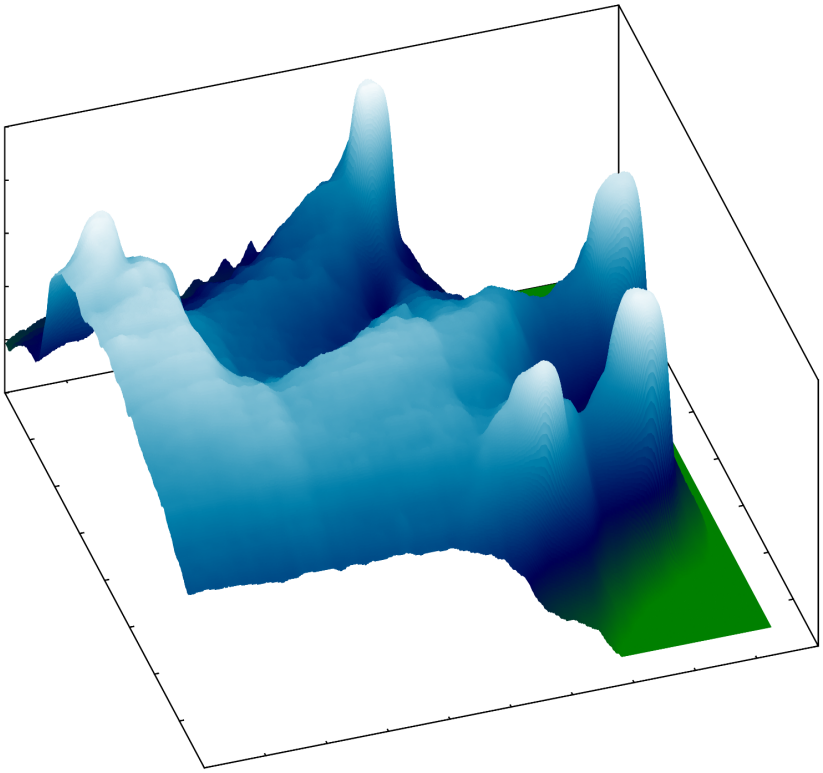
CHALMERS



A study of ambient light-independent multi-touch acquisition and interaction methods for in-cell optical touchscreens

Master of Science Thesis in the Programme Computer Science - Algorithms, Logic, and Languages

Philip Irri, Julian Lindblad

Department of Applied Information Technology
Chalmers University of Technology
Gothenburg, Sweden 2014

A study of ambient light-independent multi-touch acquisition and interaction methods for in-cell optical touchscreens

PHILIP IRRI,
JULIAN LINDBLAD

Technical report number 2014:110
ISSN number 1651-4769

Department of Applied Information Technology
Chalmers University of Technology
SE-412 96 Gothenburg, Sweden

Cover figure: Optical sensor data of a hand touching the touchscreen prototype visualized in 3D

Department of Applied Information Technology
Gothenburg, Sweden October 2014

Abstract

This thesis describes an in-cell optical touchscreen, a technology that makes use of light sensors embedded in the LCD layer. The advocates claim decreasing production costs in relation to display size, minimal affect on product design, and unlimited multi-touch functionality, as well as the possibility to acquire 3D spacial-temporal coordinates. The users would also be able to interact not only with their fingers but with all kinds of physical objects. This could challenge the well-established capacitive touch method and revolutionize the way we interact with future touchscreen devices. However, there are still many technical difficulties that needs to be solved before this could be a reality. The major obstacles are ambient light vulnerabilities and dependency on what image is displayed on the screen.

This thesis investigates these matters using an in-cell optical touchscreen prototype developed at Semiconductor Energy Laboratory. The thesis compares this device with other similar hardware and their way of solving the above stated problems. Furthermore, theory for touchpoint acquisition using image processing are explained, and a new algorithm is proposed. The proposed algorithm utilizes an estimation of the ambient illumination and takes the displayed image into consideration in order to calculate touchpoints. Furthermore, other interaction methods such as finger rotation and height above the screen are investigated.

Finally, an empirical evaluation was conducted to evaluate the accuracy and reliability of the proposed algorithm.

Keywords: in-cell, optical, touchscreen, In-Ga-Zn-Oxide (IGZO) technology, image processing, feature extraction, touch detection

Acknowledgements

Many people have been involved in our thesis work that we would like to thank for helping us.

First of all, we would like to thank Dr. Shunpei Yamazaki for making this internship in Japan possible. It has been a very valuable experience for us. We would also like to thank Edvard Fleetwood at the Sweden-Japan Foundation for offering us this internship position as well as initiating the contact with SEL. We want to thank Yoko Otake and Mika Tatsumi for all the help we received regarding the organization of the internship as well as help with any other daily obstacles that we encountered while living in Japan.

A very special thanks to our supervisors in Japan, Jiro Imada, Katsuki Yanagawa, Takashi Igarashi and Yuji Iwaki. Their help in this project was invaluable and have always been available for us. A special thanks also to our supervisor Morten Fjeld in Sweden at Chalmers University of Technology for all the valuable information about image processing and research studies in general. Another thanks goes to Alexandru Dancu, who found good references for the thesis.

Furthermore, we want to thank Hikaru Tamura for helping us getting deeper understanding in the device we worked with as well as the original implementation. We also want to thank Isamu Shigemori who helped us with the writing of the conference paper and we also want to thank Kiyoshi Kato who provided us a great review of the algorithm when writing the patent application. Finally, we would like to thank all of our coworkers at the ISG group at SEL for giving us a great welcome as well as for participating in our in the evaluation of our algorithm.

Philip Irri, Julian Lindblad
Kanagawa, Japan, August 2014

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This Master's thesis was written in conjunction with a six months long internship at the Japanese research company Semiconductor Energy Laboratory (SEL) where we were given the opportunity to do research on state of art technology of touchscreens.

The subject of this report is in the field of image processing and software development for touchscreens. During the period of this internship a patent application and a research paper were written and submitted (see Appendix C for more details).

1.1 Semiconductor Energy Laboratory

SEL is a Japanese research company founded by Dr. Shunpei Yamazaki in 1980 and its research started with solar battery technology. SEL has since expanded their area to include research about LCDs, organic EL displays, thin film integrated circuits, and a new field of electronics involving oxide semiconductors. As for SEL's income, they use an unique business model that relies on obtaining and licensing intellectual properties, a teaching taught by Dr. Yamazaki's tutor Dr. Yogoro Kato [Semiconductor Energy Laboratory Co., Ltd., 2013]. By 2011, Dr. Yamazaki still holds Guinness world record as the man holding the most patents: 6314 [Guinness World Records Corporate, 2011].

Corporate Name	Semiconductor Energy Laboratory Co., Ltd. (abbr.: SEL)
Location	398 Hase, Atsugi-shi, Kanagawa, 243-0036, Japan
Telephone	+81462481131
Homepage	http://www.sel.co.jp/en/index.html
Foundation	July 1st, 1980
President	Shunpei Yamazaki
Number of Employees	710 (as of April 1st, 2013)
Business	Research and development of crystalline oxide semiconductor thin film transistors and integrated circuits, LCDs and ELDs, rechargeable batteries, and organic EL lighting. Patent of inventions and exercising of patents rights.

Table 1.1: SEL's corporate profile

1.2 Background

Touch technologies have been around for a long time, with concepts such as multi-touch and hand-based gestures (such as pinching [Krueger et al., 1985]) dating back to the beginning of the 1980s [Lee et al., 1985] [Buxton et al., 1983]. It was not until 2007, however, when Apple Inc. announced and released the iPhone that touch enabled devices would become a big part of our daily life and set the standard on how we interact with mobile devices. When Microsoft two years later released Windows 7 with touch-support, multi-touch became a "must have" characteristic of many consumer touch devices [Colegrove, 2010].

Touch sensing devices can be implemented using several different techniques, all with their upsides and downsides. The most common technologies today are based on resistive and capacitive types which detect input from pressure and electromagnetic fields respectively. However, these types of techniques do not scale well in size and result in higher production costs and insufficient performance [Tamura et al., 2011].

To address this, SEL is experimenting with liquid crystal displays (LCDs) and electroluminescent displays (ELDs) with touch capability using optical sensors embedded in the pixels, so-called *in-cell optical* touchscreens [Tamura et al., 2011]. This technique, also known as *light sensing*, uses shadows casted on the display and the light reflected from pointing devices in order to calculate its touchpoints. The technique is explained in more detail in Section 2.2. Similar hardware with light sensing capability have been researched and developed before, but SEL's prototype touchpanel is made by oxide semiconductor (OS) field effect transistors (FETs) which enables the photo-transistors to use a *global shutter* method to capture the frame (explained more in Section 3.1)[Tamura et al., 2011]. In contrast to capacitive and resistive sensors in which the user has to physically touch the screen, optical sensors are capable of introducing new intuitive ways of interacting, such as gestures hovering above the screen and fingerprint scanning [Brown et al., 2007]. Moreover, the sensor is not limited to detect only pointing devices such as fingertips and styluses, but also whole physical objects.

1.3 Research problem

Although an optical sensor enables non-touch functionality for the display, compared to many existing technologies used for touch input, the optical sensor is highly sensitive to the surrounding environment. If implemented in a mobile device, the user might carry this to any kind of environment which might be constantly changing. Given an environment with low ambient lighting and a dark image on the screen, sensing necessary touch information is impeded. Making touch/non-touch input robust at any moment is the most difficult challenge for using optical sensors.

Furthermore, in-cell optical touchscreens are still a minority in the market and it's capabilities are not yet fully uncovered.

Another issue SEL wanted to be investigated, which is general for all touch screen devices, is that *touch* might be perceived differently from person to person. For example, under development, the touch input might work very well by the people who is developing the device. However, when tested by other people, they might perceive touched areas differently. One user might press with only the tip of the finger and expect the touch point to be directly under the touch area, whereas another user might press with a much larger area and expect it to be directly above the touch area.

1.4 Aim

The fundamental aim of this internship and report, is to evaluate and solve the given problems in Section 1.3. First of all, the information obtainable in various lighting conditions have to be examined from which a new method for detecting touchpoint can be developed and tested.

Secondly, regarding the second problem in Section 1.3, this report examines the possibilities of various methods to determine the intended touch point by the user. Using optical sensors, the potential possibility of measuring distances to the pointing device could make it possible to determine 3D spacial-temporal coordinates but also help obtain the orientation of the pointing device, which in turn might help deciding the touchpoint with higher accuracy.

Moreover, the report also examines other possibilities of human-machine-interaction that in-cell optical sensors enables. Except for simple multi-touch input, optical sensors make it possible to have other non-touch input, such as various gestures hovering above the device. Such gestures could be simple swipe movements, or more complex gestures such as pinch and rotate movements. Furthermore, since optical sensors not only are limited to detect simple pointing devices, new application areas will be investigated.

Finally, throughout the whole research process, the overall performance of the method should always be considered. The prototype has a refresh rate of 60 frames per second, and thus the runtime should have an upper limit of 1/60th of a second.

1.5 Limitations

Although the research should always have performance of the algorithms in mind, it is not of interest to optimize on a too low level, as this is a topic of its own. Furthermore, optimizations via parallelizations are also not considered, as this is a rather trivial optimization technique and not very interesting for our research, nor should it be assumed that the final implementation of the algorithm will utilize hardware capable parallelism.

Because the work is only performed on a prototype which is not portable, the work is only assuming in-door environments. Therefore, it is limited to homogeneous constant lighting.

Chapter 2

Previous work

Today's major touch technologies can be divided into many different categories, with further sub-technologies. The two major categories are resistive and projected capacitive. Other popular technologies are surface capacitive, infrared, optical, acoustic wave, in-cell, among others. These techniques are summarized in Table 2.1 and further explained in this chapter.

One interesting fact to notice about touch sensing in today's technology is that touch is actually an indirect measurement. That is, the information of *touch* can only be measured by measuring some other information. For example, capacitive sensors measures either change in capacitance or voltage in order to detect touch, resistive uses voltage, vision-based uses changes in image and embedded light-sensing uses presence and/or absence of light. The fact that touch is an indirect measurement is the reason to why there are so many technologies and researchers try to discover and experiment with new ways of detecting touch [Walker, 2013].

2.1 Conventional touch technologies

A large variety of touch technologies exist today other than in-cell optical that the prototype for this research is using. Today, capacitive and resistive touch technologies are the most popular, however other technologies that exploits completely different physical phenomenon also exist. All these technologies are briefly explained in this section.

Summary: Touch technologies		
Technology	Advantage	Disadvantage
Projected capacitive	Multi-touch	High cost
Surface capacitive	Touch sensitive	No multi-touch
Resistive	Low cost	No multi-touch
Acoustic	Durability	Soft touch objects
Camera-based optical	Very simple production	High profile
Traditional infrared	Reliability	High cost
In-cell optical	Integration	Ambient noise

Table 2.1: A summary of the major advantages and disadvantages that are discussed in this chapter [Walker, 2013].

2.1.1 Capacitive

Projective capacitive (pro-cap) touch technology is, as of today, the most common technology used in mainstream products [Walker, 2013] and was popularized by Apple with the introduction of the iPhone to the market [Colegrove, 2010]. The basic principle of pro-cap is to measure the differences in capacity and when a human comes into contact with the electrode, the human body's capacitance affects the self-capacitance of the electrode. The electrodes are then layered out in a grid in either one or two layers. Furthermore, pro-cap technologies can be divided into two groups, *self-capacitance* and *mutual-capacitance*. Self-capacitance systems measures whole rows and columns which can create ghost points and thus can only support limited multi-touch. However, mutual-capacitance measures all the intersections, and thus can achieve true and unlimited multi-touch. Other advantages with projective capacitive touch devices are very high optical quality, i.e., very high see-through of the layers, as well as high durability. On the other hand, it is noise sensitive, anything that generates an electromagnetic field can cause disruptions. It is also limited to touch from human skin, or other objects specifically designed for capacitive touchscreens, and can not be used with other objects. Finally, the production cost for projected capacitive touchscreens are still high, however it is steadily decreasing.

Another type of capacitive touch technology is *surface capacitive* touchscreens. The principle is the same as pro-cap; that is, a human's capacitive property results in the possibility of detecting a touch point. One sensor is positioned at each corner of the display, and when the user interacts with the display, the current is changed and detected by the sensors. The touch position can then be calculated from these changes [Walker, 2013]. They can be made durable and with high sensitivity, however they are just like pro-cap touchscreens limited to human touch. Furthermore, compared to pro-cap, it can only handle single touch and the optical quality is lower. It might also need recalibration during the usage.

2.1.2 Resistive

Resistive touchscreens are both simple and is a cheap technology. It also has a low power consumption and is widely available. The display glass and an outer film are both coated with a conductive layer that are separated with a thin space of air. Then, by applying an voltage on one layer, and measuring it on the other, a touch position can be acquired. Compared to capacitive touchscreens, resistive can be operated with any type of pointing device. However, it is fragile and has a poorer optical quality and has not support for multi-touch [Walker, 2013].

2.1.3 Acoustic

Surface acoustic wave is an inventive method that uses *transducers* on each axis of the screen to generate ultrasonic waves over the display. If the user interacts with the display, it will disrupt the wave which can be measured by the controller and the touch position can be registered [Walker, 2013]. Since it has a high optical quality, it works with many pointing devices and is very durable, this technology is often deployed in machines that are publicly available such as ATM machines and ticket machines. However, it does not support multi-touch, requires a sound absorbing pointing device, as well as a nonzero activation force.

2.1.4 Optical

There are many different optical touch detection technologies, however, what they all share is the fact that they are using light sensors in some way for touch detection. Typically, the user operates on a 2D surface that one or more cameras records which it uses for processing.

Ever since the 60's researcher have tried to make use of *frustrated total internal reflection* (FTIR) [Wyman, 1965]. Han [2005] showed a way to construct a very simple and low-cost touchscreen that utilizes the effects of FTIR by putting a infrared sensitive camera behind a sheet of acrylic. On the sides of the acrylic, infrared light is let in and total internal reflection prevents the light from escaping the material. When a user touches the plane, however, the light becomes *frustrated* and reflects from the finger out through the acrylic and into the camera. A touch can thus be detected by analyzing the camera footage with simple image processing, explained in Chapter 4. The technique has been further improved to calculate other parameters than just the position on the 2D plane. In the work by Tulbert [2005], parameters such as normal force, velocities, height, finger angle and orientation could be calculated with errors of 2% or less.

Another type of optical touchscreen devices uses embedded infrared transmitters and receivers in the border of the display, which can be positioned in various patterns. This type of touch technology is refered to as *traditional infrared* by Walker [2013]. The most simple pattern is to just have the transmitters and receivers on opposite sides of the display, making up X- and Y-axis, and let them shoot straight beams of infrared light. The light is then detected on the opposite side and if there is a shadow, a touch is detected. This simple design is however not able to detect more touch points than two, as it introduces ghost points. Other patterns use transmitters that have a higher degree of spread and/or interlaced transmitters with receivers. These can achieve an higher number of touch points. Similar setup can also be achieved with regular cameras positioned in the corners of the display.

Stereo cameras, mounted either above or under the display, can also be used to detect touch. A camera setup like this was used together with machine learning methods as well as a geometric finger model to produce a high precision multi-touchscreen [Agarwal et al., 2007].

Yet another way is to position an infrared light emitting source together with the infrared camera behind the display. This was used in the first version of Microsoft Surface as an attempt to create a device with a *tangible user interface* (TUI) that would bridge the physical and virtual worlds [Dietz and Eidelson, 2009]. A projector was mounted inside the device that projected the image on a diffuser to show the screen. Five IR cameras and an IR light emitter was also mounted inside the detected IR light reflecting when the user touches the device.

2.2 In-cell optical touch technologies

The focus of this thesis is to study an in-cell optical touchscreen. The technology is very similar to vision based touch technologies, in that the system often receives an image of the user from the display's point of view. Just like vision based systems, a light source can be used to emit infrared (IR) or plain visible light from the display. However, instead of using a regular camera positioned behind the display pane, the in-cell touchpanel, as the name implies, uses photo diodes embedded within the pixel layer.

Initially, the use of in-cell technologies appears to have many good prospects. As the sensors are embedded with the pixels, there is no extra size of the display, and the display does not affect the design of the final product. Since every sensor in each pixel are measured completely independently, theoretically any number of touchpoints can be obtained with very high accuracy and performance [Walker and Fihn, 2010]. Lower parts count, lower production cost as well as better performance on larger touchscreens are the major advantages of using in-cell display technology [Tamura et al., 2011, Brown et al., 2010].

In spite of the above mentioned advantages with in-cell optical technology, it still have some weaknesses. Due to the low transmission rate for off-state liquid crystals (i.e., the displayed image is black), the optical sensors will not be able to obtain good enough measurement from

the reflected backlight if there is a lack of ambient light [Tanaka et al., 2011]. This is often referred to as the black (display-)image problem. A common way to overcome this problem is to increase the amount of infrared light emitted from the backlight. Also, by using an IR pass filter to only detect IR light, the dependency of the image displayed on the screen is removed. Increasing the amount of backlight will however negatively affect the power consumption.

Furthermore, in situations where the image is non-black, very high or very low ambient light can cause problems. In bright environments, it is difficult to distinguish the shadow from the hand hovering above the display and the finger which is actually touching the display. Similarly, in dim lit environments, it is difficult to distinguish between the reflection from the backlight on a finger and the ambient light falling on the screen. Furthermore, placing light sensors in every pixel in the display turned out to consume too much of the pixel's aperture, in other words block to much of the emitted light. Also, the time and power to process data for too many sensors greatly increases [Walker and Fihn, 2010].

The first big announcement of putting light-sensing elements into the pixels was made by Toshiba Matsushita Display Technology Co., Ltd. (TMD) in 2003. Two years later a prototype using this technology was developed that detected touch from the shadows of the pointing devices [Buxton et al., 2007]. TMD was also first to describe the process of using both reflected light and shadows casted from pointing devices, and how to switch between the input automatically. The first commercially available product using this technology was Sharp's netbook PC-NJ70A which was released in 2009. However, according to Sharp the product was only a technology experiment [Walker and Fihn, 2010].

2.2.1 Similar devices

Boer et al. [2003] at Planar Systems produced an 240 × 240 pixel 3.4 inch TFT LCD with an embedded 60 × 60 array of a-Si photo sensors. It was able to obtain the 2D-coordinate of a shadow cast by a finger provided ambient lighting greater than 50 lux. In darker conditions the touchscreen was unable to operate in the shadow mode. However, they noted that the problem could be solved using a backlight to detect finger reflections. They also stated that by using a photo sensor array in the pixel layer possible to measure the ambient light intensity which in turn can be used to control the back-light intensity without the need of an external photo-sensor.

The following year, the same team (Abileah et al. [2004]) developed a 14.1 inch 1024 × 768 pixels AMLCD with an embedded sensor array consisting of 256 × 192 photo sensors. This touchscreen had improved optical sensor performance and special electronics for noise reduction. Moreover, they used a two-frame average to further reduce noise and improve performance. In order to operate in dark lighting conditions they installed a edge lamp in the touchscreen. It was turned on when the average sensor reading was below a specific illumination level, and thus fingers in contact with the screen could reflect the light back to the sensors. Both devices developed by Planar Systems could run its processing on programmable hardware which resulted in low cost and high performance.

Another similar technology is ThinSight, an optical sensing system developed by Hodges et al. [2007] at Microsoft Research. This out-cell technology is based on a flat 2D grid of 15 × 7 IR retro-reflective[1] optical sensors and emitters placed behind the LCD panel. In spite of the low number of sensors, fingers and hand within proximity of the screen are clearly identifiable, however smaller objects can disappear on occasion when they are between the sensors. Reliable touchpoints can be detected within 1 mm above the panel. It can operate from complete darkness to direct sunlight because of the nature of the retro-reflective IR sensors. The power consumption of the IR light however makes this technique impractical for mobile devices.

[1] A device that reflects light back to its source with a minimum of scattering.

Brown et al. [2010] at Sharp Laboratories in collaboration with Semiconductor Energy Laboratories presented a 2.6 inch AMLCD with in-cell photo sensors. It works the same way as previous mentions devices; reflected light is detected by the sensors and generates an image which can be processed to extract features such as coordinates. This device's photo sensors however, are set to look in four orthogonal directions (up, down, left, and right). As a result it is possible to obtain not only the 2D position of interaction devices, but also the height (up to 20 mm) above the screen. This is done by examining the relative displacement of the object's planar coordinates in each directional sub-image. The height is then calculated by triangulation. By using four directions for the sensors, it is possible to detect robust 3D coordinates even in corners of the screen.

Sharp has also developed another LCD with optical input functionality [Tanaka et al., 2011]. They are using an IR backlight sensing method, which solves the "black display problem" described above and in order to operate under a wide range of ambient illumination without significantly increasing the power consumption a *backlight substraction scheme* is introduced. The IR backlight is synced with the photosensing processes, and the optical noise is removed by subtracting the sensor data when the backlight is turned off with the sensor data when the backlight is turned on. The device has been tested and can operate from zero to 70,000 lux.

Microsoft PixelSense (previously know as Surface) is a table-top multi-touch system which can interact with the users in various ways. The device is designed as a table with a display on the top, which the users can interact either with pointing devices or their own hands.

Microsoft developed two versions of the PixelSense, the first version utilized a vision based technology as explained in Section 2.1.4 [Friedman, 2007]. However, that device was very thick, reaching from the floor up to the table's top. The second version was improved to only be 4 inches thick by switching to an in-cell solution[Microsoft Corp., 2012] called SUR40, produced by Samsung Group [Microsoft Corp., 2012]. However, even though it only uses IR light for touch processing, according to Walker [2012] it is still very sensitive to external light that interfere with the touch detection.

Chapter 3

The in-cell optical touchscreen prototype

The prototype, which was developed in 2011, device is a six-inch oxide semiconductor FET LCD with 768 (H) × 1024 (V) pixel resolution updated at 60 frames per second (FPS) with an embedded light sensor at every fourth pixel. Using a global shutter, the device can capture two sensor images per frame; one image is taken with the backlight turned on, whereas the other with the backlight turned off. These two images (see Figure 3.1) are sent to a host system and processed for touchpoint detection. For a detailed hardware specification list see Table 3.1.

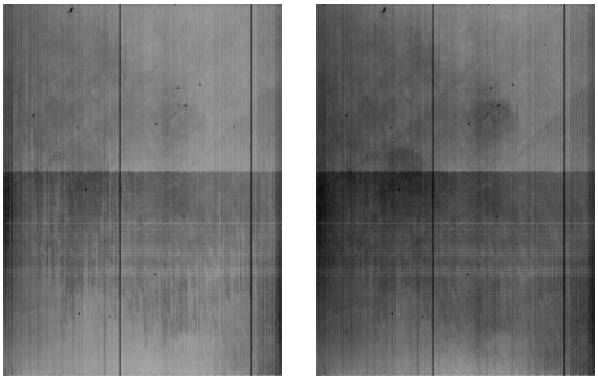
Hardware specification table	
Display type	In-Ga-Zn-Oxide (IGZO) FET LCD
Display size	6-inch (diagonal)
Manufacturer	Semiconductor Energy Laboratory Co., Ltd. (SEL)
Backlight	White LEDs
Display resolution	768 (H) × 1024 (V) (XGA)
Pixel pitch	0.12 × 0.12 mm
Aperture ratio [1]	42.5 %
Sensor type	In-cell photo diodes (a-Si)
Sensor resolution	384 (H) × 512 (V)
Sensor capture rate	60 Hz
Shutter type	Global shutter

Table 3.1: Hardware specification for SEL's touchpanel.

3.1 Hardware

The in-cell optical touchscreen device consists of three layers (see Figure 3.2). The bottom layer is the backlight, which consists of a grid of white LEDs which illuminates the next layer: the liquid crystal display. In front of all the liquid crystals there are color filters in a red, green,

[1] The ratio between the area that can emit light and the total area of the panel.



(a) Backlight turned on (b) Backlight turned off

Figure 3.1: Showing the raw sensor data as two rasterized grayscale images. Careful observation can reveal two fingers touching the device.

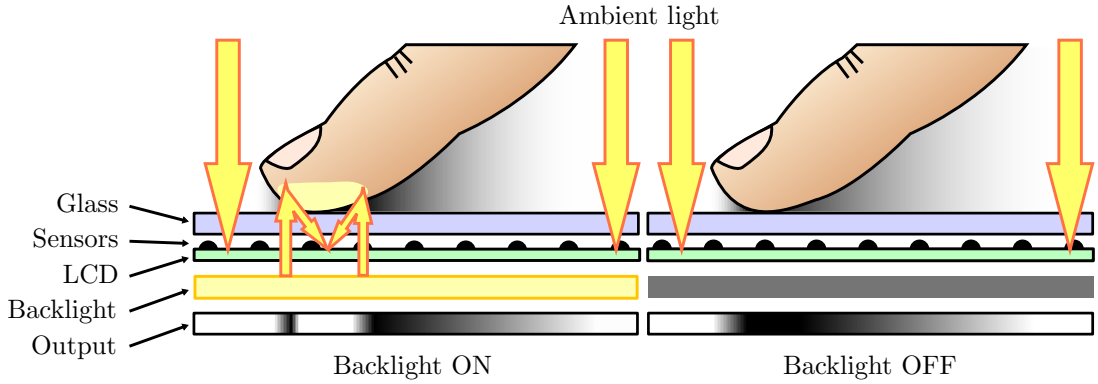


Figure 3.2: Overview of the hardware. The touchscreen is able to capture two sensor images per frame; one image is taken with the backlight turned on and one with the backlight turned off. Note how the backlight is affecting the sensor output.

and blue (RGB) pattern. Three liquid crystals, each with a RGB filter respectively is called a pixel. Depending on the current flowing through each of the liquid crystals, they will block or let through a certain amount of backlight, which will be visible for the user. In the same layer as the LCD, the optical sensors are also embedded for every fourth pixel. On top of the combined LCD and sensor layer there is a glass pane that protects the underlying structure from taking damage (e.g., too high pressure, dust, and moisture).

Global shutter

The two common techniques for controlling and reading the light sensors in optical touchscreens or cameras are by using a rolling shutter and a global shutter [Tamura et al., 2011]. Most common is the use of a rolling shutter which exposes each sensor row and then measures them sequentially. In comparison, a global shutter exposes the whole matrix of sensors simultaneously and then row-by-row, or in this case, column-by-column measures the sensors. The major disadvantage of a rolling shutter compared to a global shutter is the distortions introduced by fast moving objects.

However, the time needed for the sensors to maintain the current (i.e., the pixel intensity value) in a global shutter is usually too long when using regular low-temperature polysilicon (LTPS) FETs. This results in noisy sensor values. OS FETs on the other hand, can maintain the current for a much longer period and therefore implementing a global shutter without a special unit for holding the charge using these FETs is trivial [Tamura et al., 2011]. Since the global shutter is fast, the backlight can be turned on and back off for each frame captured. Meanwhile the sensors can be measured once for each such state without any experience of flickering. This is done both in order to enhance the reliability of the touch detection as well as to help detecting touch in environments where a shadow from the user is not available.

3.2 Properties

As described in Section 2.2, the transmittance rate for off-state liquid crystals is close to zero for visible light. In other words, this means that when the display is showing black images, nearly no backlight will be transmitted. For liquid crystal displays that do not use optical sensors such as flat screen televisions, computer monitors, and capacitive touchscreens, this is often seen as a good characteristic since it will enhance the contrast between white and black. For devices that uses optical sensors for touch detection though, this leads to a great drawback. As illustrated in Figure 3.3, when the display is showing black or dark images the liquid crystal layer will prevent the light from being transmitted, resulting in weak sensor readings.

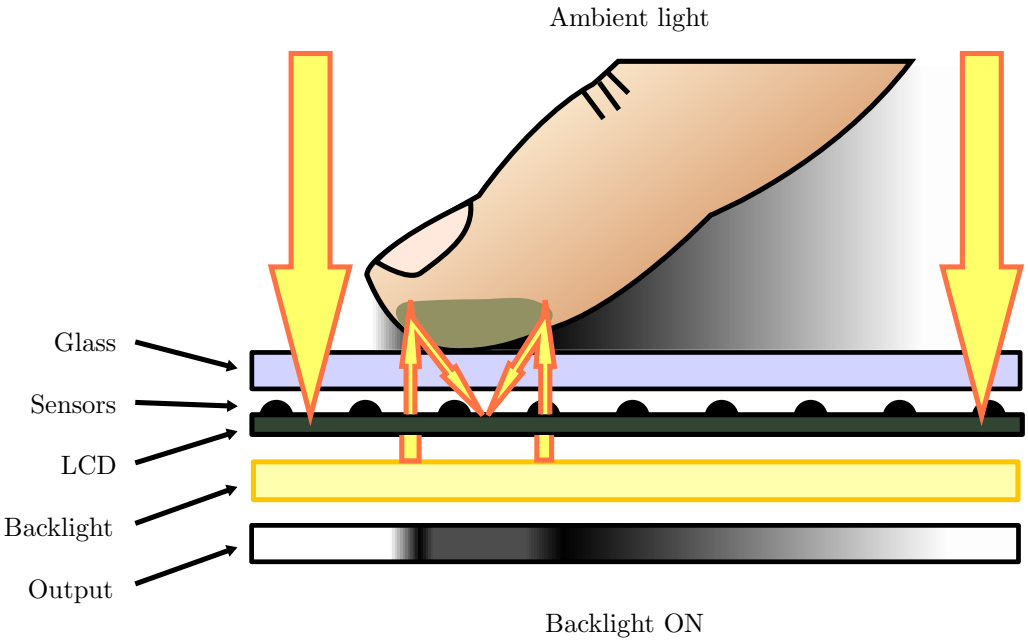
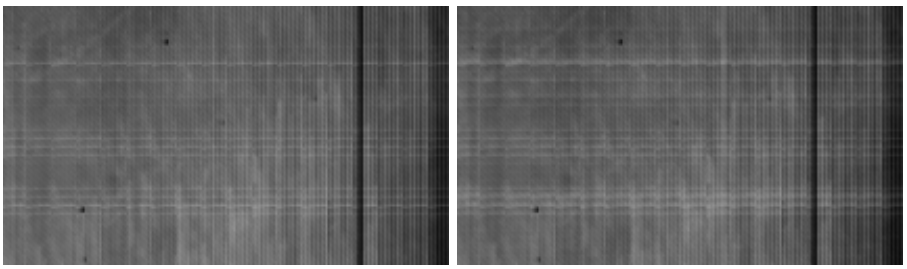


Figure 3.3: When the display is showing a black image (i.e., the liquid crystals are in off-state and do not transmit the white backlight), the light reflected on the finger will be weaker.

On the positive side, by limiting the backlight to the visible spectrum (in contrast to the addition of the infrared spectrum), the power consumption can be kept low.

Moreover, due to the fact that the touch device is basically a camera without a lens, the sensors will capture light from a wide range of incident angles, resulting in a very diffuse representation of the depth from the display pane. Consequently, this limits the maximum distance the sensors can measure objects above the screen to a few millimeters.

As stated above, the device was a prototype. The sensors of the display were sensitive and vulnerable to excessive wear. As a result, we were urged to record sensor data as video files, instead of directly interacting with it. Furthermore, the prototype was very sensitive of higher temperatures. See Figure 3.4 for a comparison of how the temperature affects the sensors data. At about 30 degrees Celsius and above, the sensor started to behave strangely, finally ending up in an inoperable condition where most of the sensors outputted their maximum value, resulting in a nearly white image as can be seen in Figure 3.5.



(a) Shortly after upstart. (b) After four hours of usage.

Figure 3.4: Two comparison image of the sensor data taken at different times. After four hours of usage both the device and the room got warmer as there was no air-conditioner. Notice how the horizontal lines are much thicker as seen in (b).

Furthermore, even though the device uses a global shutter and the two input images can be taken within a very short interval, it is still sometimes too slow, which can be seen in Figure 3.6. If the user moves her hand too fast, the spatial difference between the two images get too large which results in an error that manifests itself as a *tail* for the touchpoint which may disrupt the touch detection.

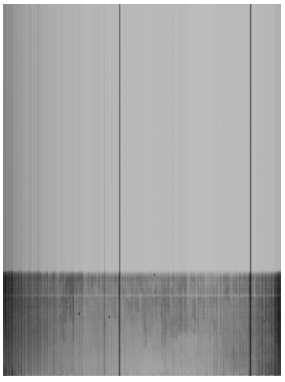
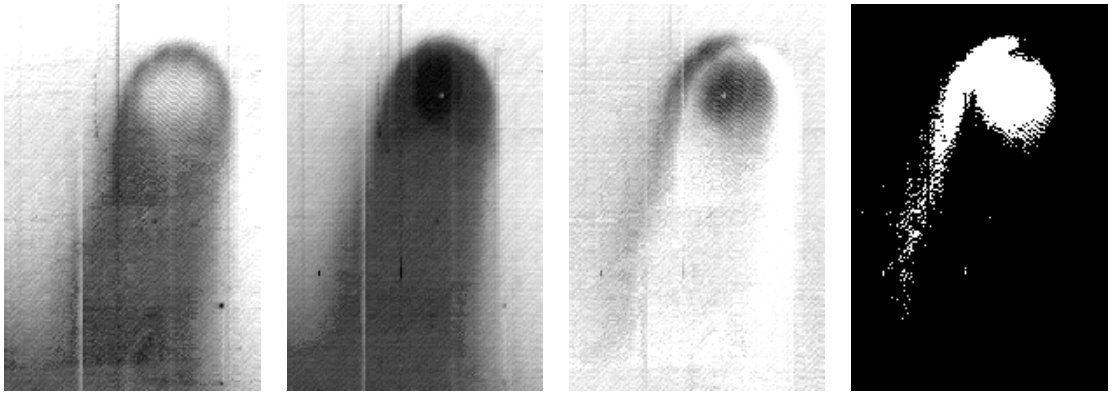


Figure 3.5: An example of sensor image when the device has become overheated.



(a) Calibrated input image with backlight turned on.

(b) Calibrated input image with backlight turned off.

(c) The difference between the two input images.

(d) The binary image after thresholding the difference.

Figure 3.6: The images shows the problem with too quick motions on the device. The images taken with backlight on and off respectively can not be taken at the exact same time, and if the movement is fast enough, an error manifests itself in the difference shown in (c). This is especially clear in (d) which shows a very clear *tail*.

Chapter 4

Theory

This chapter will cover the theory of the work. The basis of digital image processing will be explained, followed by some common implementation techniques for touchpoint recognition and acquisition. The last section will go through the theory behind conducting an empirical evaluation.

4.1 Digital image processing

This section will explain the basic procedures of a typical image processing system. The fundamental steps in digital image processing can be categorized as the steps in the flow chart shown in Figure 4.1 [Gonzalez and Woods, 2008]. These steps will be explained below.

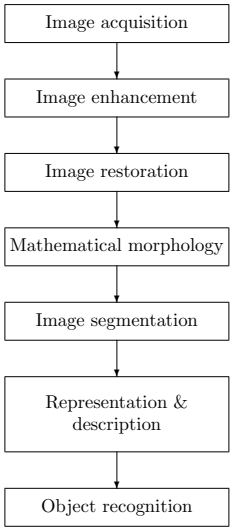


Figure 4.1: Fundamental steps in digital image processing

Image acquisition refers to the process of acquiring the raw input image, which usually is a frame from a video stream or capture device. The frame usually have been captured by analogue sensors which then is digitalized by an *analog-to-digital* converter (ADC). Image transformation such as scaling can also be applied in this step.

Image enhancement is the process of bringing out obscured details or highlight features of interest in an image. A common enhancement technique is to change brightness or increase the contrast of an image.

Image restoration aims to improve the appearance of degraded images that might be corrupted or noisy. The techniques are based on mathematical or probabilistic models and can repair images from defects such as motion blur, noise, unfocused, and non-linearity of electro-optical sensors [Sonka et al., 2007]. A common improvement technique to remove noise is to apply a filter, typically a Gaussian filter, which replaces each pixel value with the average value of the surrounding neighbouring pixels.

Morphological processing is a theory and technique based on set theory and shares similarities to convolution. *Mathematical morphology* is often applied to binary images and typically makes use of two basic operations, dilation and erosion, to simplify the images and preserve the main shape and characteristics of objects in them [Sonka et al., 2007]. The mathematical definition of the dilation operation is:

$$X \oplus B = \left\{ \mathbf{d} \in E^2 : \mathbf{d} = \mathbf{x} + \mathbf{b} \text{ for every } \mathbf{x} \in X \text{ and } \mathbf{b} \in B \right\} \tag{4.1}$$

where X is the set of all pixels with value 1, B is the *structuring element* that is moved across image X, and $\mathbf{d}$ is all the pixels with value 1 that creates the new image in the 2D Euclidean space E^2. The operation, if used with an isotropic structuring element (i.e., it behaves the same way in all directions), is also called fill or grow since it expands the characteristics in the image, and also fills holes and narrow peninsulas.

Similarly, erosion is defined by

$$X \ominus B = \left\{ \mathbf{d} \in E^2 : \mathbf{d} + \mathbf{b} \in X \text{ for ever } \mathbf{b} \in B \right\} \tag{4.2}$$

using the same notations as explained above. Conversely, erosion is called shrink or reduce if used with an isotropic structuring element and is used to eliminate small defects and make characteristics in the image thinner. Already with these simple operations, relatively advanced results can be given; an easy way to obtain the contours of an image is to subtract the eroded image from its original.

By performing the above operations one after the other, two new useful morphological transformations are created: opening and closing.

Using the same notations as above, opening is defined as

$$X \circ B = (X \ominus B) \oplus B \tag{4.3}$$

, in other words erosion followed by dilation. The opposite, dilation followed by erosion is called closing and is defined by

$$X \bullet B = (X \oplus B) \ominus B \tag{4.4}$$

Opening and closing with an isotropic structuring element is used to eliminate image details smaller than the structuring element itself, while keeping the global shape of the objects in the image intact.

Moving on, *image segmentation* divides an image into its constituent parts or objects, which then will be easier to analyse in further steps. The basis is to assign a label to every pixel such that pixels with the same labels share certain visual characteristics. One of the most simple image segmentation procedure is thresholding, which labels the pixels black or white depending on a threshold value.

The next step in the image processing system is *representation and description*. Given the pixel data for each segment from the step above it decides whether the data should be

represented as a *boundary* or *complete* region. Boundary representation is suitable for external shape characteristics (e.g., corners and inflection), whereas complete region representation is suitable when the internal properties of a shape is wanted (e.g., texture and skeletal shape). The description part, also called feature selection, extracts attributes for the segmentation which then is used for differentiating one class form another.

Finally, *object recognition* is where all segmented parts from the image is categorized (based on its descriptions) into objects.

Following the fundamental steps in Figure 4.1, the process of acquiring touchpoint information from raw touchpanel data can be made clear and structured.

4.2 Touchpoint calculation

The process of acquiring touchpoint features such as 2D coordinates from the raw input data can follow the general steps previously described. When it comes to object recognition, especially in non-vision-based touch acquisition systems, it is simpler and more straightforward compared with a real world image taken by a camera [Wang et al., 2008] mainly because of the lack of a background in the raw image data (c.f., the techniques in Section 2.1), but also that segmented blobs always will be represented as boundaries and generally will be assigned the same description, i.e., touched areas (whereas in real world photos blobs might represent objects such as people, trees, cars, clouds etc., that might need to be classified differently). The need for decision-making classification is obsolete in touchpoint recognition.

Additionally, a complete touchscreen system usually includes a background subtraction scheme, blob extraction, feature extraction, followed by a tracking system [Abileah and Green, 2007, Wang et al., 2008, Han, 2005]. See figure Figure 4.2 for a general image processing system specialized for touchpoint acquisition.

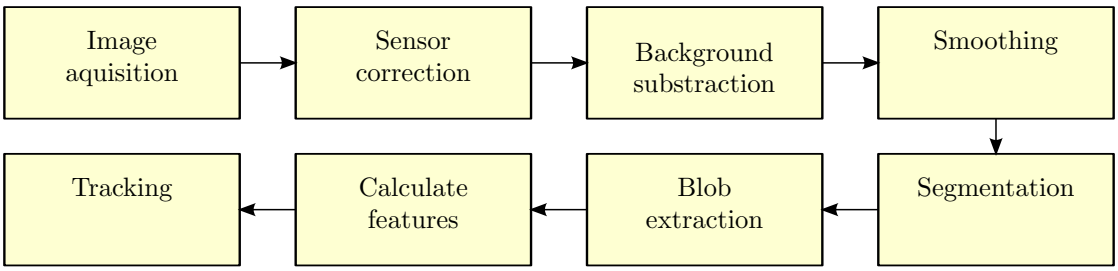


Figure 4.2: Showing the steps in a common touchpoint acquisition system.

In a patent filed by Apple Inc. in 2004, a basic procedure turning capacitive touchscreen raw data into coordinates was shown [Hotelling et al., 2006]. The received data is first filtered to remove noise such as scattered single or unconnected points. It is thereafter used to generate *gradient data* which indicates the topology of the connected points, blobs. Given the gradient data as input, the boundaries for the touched regions are then calculated using a watershed algorithm. In the Watershed algorithm the gray-scale image values are interpreted as elevation in a topographical relief. The basic idea is that if a drop of water would fall somewhere on the relief, it flows down a path reaching the local minimum. From this the image can be segmented by the edges it detects [Vincent and Soille, 1991].

A study by Wang et al. [2008] uses a vision-based multi-touch device in which they make use of a *find contour function* provided by the OpenCV library as their segmentation step, in order

to obtain the contours of the touched areas from the processed image. According to the study the contour function provides good results and have an acceptable processing time.

Blob extraction is the procedure of extracting the regions given by the image segmentation step. One common way to achieve this is the *connected-component labeling* which was described by Rosenfeld and Pfaltz [1966]. The definition of the problem is as follows: let I be a binary image and F and B two subsets of I corresponding to the foreground and background pixels respectively. A connected component C of I, is a subset of F of maximum size such that all pixels in C are connected. Two pixels, P and Q, are connected if there exists a path of pixels between them. Connectivity is defined by a pixel's neighbourhood and can be connected 4-way or 8-way. Given a binary image, the connected-component labeling will return a new image in which an unique label is assigned every group of connected pixels [di Stefano and Bulgarelli, 1999].

There are two major implementation methods for this algorithm: recursive and two-pass method [Shapiro and Stockman, 2001]. The recursive method runs in one pass. Given a binary image stored as an 8-bit array, first negate it such that the foreground pixels represents unlabeled area (-1). Then raster-scan iterate over each pixel in the image. If it finds an unlabeled pixel, label it, and recursively find its unlabeled neighbours until that area is covered. Increase the label count and continue the iteration. See Algorithm 1 for pseudo code.

Algorithm 1 Recursive implementation of connected-component labeling

```
1: function RECURSIVECONNECTEDCOMPONENT(image)
2:     label ← 0
3:     labeled_image ← negate(image)
4:     for all position in image do
5:         if image.at(position) = −1 then
6:             label ← label + 1
7:             SEARCH(image, label, position)
8:         end if
9:     end for
10: end function

11: function SEARCH(image, label, position)
12:     image.at(position) ← label
13:     neighbours ← getNeigbours(image, position)
14:     for each neighbour in neigbours do
15:         if image.at(neighbour.position) = −1 then
16:             SEARCH(image, label, neigbour.position)
17:         end if
18:     end for
19: end function
```

The classical two-pass method consists of two subsequent raster-scans of the image I. The first scan assigns a temporary label to each pixel in F. This temporary label is based on the values of its already visited neighbouring pixels. For 4-way-connectivity the already visited neighbours are north and west of the pixel, and for 8-way-connectivity north-west and north-east are also included. Let N be the set of a pixel's already visited neighbours, and $N_f = N \cap F$, in other words the set of that pixel's already visited neighbours that belong to the foreground. If N_f is empty then assign the pixel a new unique label. If all pixels in N_f share the same label, then assign that label to the current pixel. If two pixels in N_f have different labels then the current

pixels, assign either of them to the pixel and then register that all the labels in N_f actually are equivalent. A second scan is then run over I to replace each temporary label with a correct label with regards to the registered equivalences from the first scan.

When all the blob areas and their locations have been found, a common way of obtaining the coordinates of the touched area is by using a centroid calculation. The 2D position is defined as

$$x_{center} = \int_A x \, \mathrm{d}A/A, \quad y_{center} = \int_A y \, \mathrm{d}A/A \tag{4.5}$$

where A is the area of the blob [Wang et al., 2008].

4.3 Evaluation

To evaluate the device and the algorithm, a method to test the performance is needed. This is done by setting up an *empirical evaluation* that lets people not familiar with the technology operate the device. How well they perform is then measured. This section will describe the general theory that was used as a basis for the evaluation as described by Preece et al. [2002].

4.3.1 Pilot tests

Before running the final test, which might involve many test subjects as well as many hours of days of preparation, it is valuable to run the test on a smaller group of test subjects to evaluate the test itself. This is a cheap and an efficient way to capture practicality flaws as well as finding out unnecessary phases of the test.

Since the test executioners are well versed in the field of the test, they might very well assume the subjects have knowledge they do not have. This means that practicalities such as test instructions or questions might be difficult to understand or easy to misinterpret for the subjects. They might also find out that some steps are unnecessary or take too long time. If the tests includes questionnaires, badly formulated or meaningless questions might also be found.

In theory, any number of pilot tests can be run in order to refine the final test, as long as the resources permit.

4.3.2 DECIDE-framework

As a tool for helping the planning and execution of an evaluation, the *DECIDE* framework can be used as a guiding tool. It consists of six components [Preece et al., 2002]:

1. Determine the overall *goals* that the evaluation addresses.
2. Explore the specific *questions* to be answered.
3. Choose the *evaluation paradigm* and *techniques* to answer the questions.
4. Identify the *practical issues* that must be addressed, such as selecting participants.
5. Decide how to deal with the *ethical issues*.
6. Evaluate, interpret, and present the *data*.

4.3.3 Evaluation paradigms

There are different methods for evaluation that measures different properties and are more suitable for certain situations. Preece et al. [2002] names four different categories of evaluation paradigms that have different applications.

"Quick and dirty"

As the name implies, the method emphasises evaluation permit and without exhaustive preparation. They are meant to be made quickly and give feedback quickly and can be executed at any stage in the development. The feedback can be given as informal conversations rather than carefully written documents.

Usability testing

Usability testing on the other hand is a much more rigorous, controlled, and structured evaluation paradigm compared to the "quick and dirty" approach. The test subjects are asked to perform a specific set of tasks in a specific order in a laboratory-like controlled environment. The subject is given a detailed description on how the tasks should be performed and during the execution, everything is recorded on video which can be used to calculate times, errors or other detailed analysis. In general, such quantitative techniques are used but qualitative techniques are also sometimes present. For example, user questionnaires and interviews can also be conducted in order to take the subject's personal opinions into account. The test is performed using either a prototype or a product.

Field studies

In contrast to usability testing, field studies are performed in environments natural to the tested product and user. The purpose of the field studies is to observe the user's natural behaviours with the product. For field studies, the emphasis is on qualitative techniques rather than quantitative, such as recordings, interviews and conversation notes. The evaluators can either be an *insider* or an *outsider*, i.e., either be an participant in the test or strictly an observer of the test, and therefore measuring different aspects.

Predictive evaluation

In recent years, predictive evaluation has become a popular method for evaluation [Preece et al., 2002]. In contrast to most evaluation paradigms, however, predictive evaluation do not include any actual users. Instead, field experts are contracted that use their knowledge to evaluate a product, often by applying established heuristics. Since this method does not involve any test subjects, it is a quick and cheap method, however, it relies on the heuristics and that they are relevent for the product in question.

Chapter 5

Methodology

As Section 1.3 points out, the initial goal of this thesis was to investigate the touch detection capabilities during different ambient lighting and display image conditions. Secondly, the aim was later shifted to investigate the possibility of novel interaction methods. In the following sections the methodology and execution of the thesis work will be described.

First, the basic approach will be described following by the structure implementation of the application. After this, the calibration process for the touchscreen is explained. The next section directly addresses the first goal and a way to reach it. In its subsections experiments and an algorithm for tackling the problems is proposed. Thereafter, an evaluation of interaction methods is performed to study further possibilities with in-cell optical touchscreens. Finally, a user test was conducted in order to test the prototype and the algorithm with other people.

5.1 Approach

This section describes the tools and application used in this study as well as the general development methodology.

5.1.1 Tools

Our assigned workstations at SEL were Dell OptiPlex 9020 running 32-bit Windows 7 Ultimate. They were equipped with Intel Core i5-4570, which had four cores, each clocked at 3.20 GHz. The graphics units were basic integrated Intel HD Graphics 4600. Most of the software development were done using Microsoft Visual Studio 2010. Even though the developed application at this point only will be used internally by SEL on Microsoft Windows machines, touchscreen devices today run on many different platforms, including Android, iOS, and Windows Phone. Thus, in case if the code is required to compile to another system in the future, the code was also written using as much cross-platform libraries as possible.

Moreover, it was decided to use the image processing library OpenCV. OpenCV is a permissive open-source library that provides necessary data types and operators for image processing. It also includes convenient built-in functions such as filtering, transformations, and structural analysis. OpenCV also has the possibility to run some of the code on the GPU, thus enabling faster processing. Furthermore, the OpenCV library provides easily accessible video stream capturing functionality which makes it simple to get input from both image and video files, and camera devices such as the touchscreen hardware. The reason for choosing OpenCV as the image

manipulation library was partly because it is widely used in the field of computer vision, but also since it has a large user base with a strong community [Itseez, 2014].

In order to collaborate efficiently, Git was used as a distributed revision control and source code management system [Conservancy, 2014]. A repository for the code was set up and the base application structure was implemented together as a pair.

5.1.2 Execution

Since touchscreen hardware and image processing were a relatively new field for us, and a great part of a Master's thesis work includes literature studies, it was decided that an iterative and incremental developing methodology was going to be used. The main steps in the iteration were literature studies (including research papers, programming libraries, previous work), followed by implementation and evaluation. See Appendix A for the time plan of this thesis work.

The first task was to review the previously developed C-application that was given. This included both some image processing and specific application behaviour code. However, the given code constituted mostly of make-shift code, all written in a single source file. Also, the variable and function names, as well as the comments were all written in Japanese and it was therefore decided to rewrite the code base from scratch.

C++ was chosen as the new programming language for the application, since it is object-oriented, has good performance, and both of us had previous experience of using it in team projects. Effort was put into making the new code base both modular and fast. After finishing the code base together, the porting of the code was easily divided into their own separate blocks independent of each other, which increased the overall developing speed for the project.

5.2 Application

It was desirable to create an application that was easy to extend and to test new ideas on, quickly change algorithm-oriented parameters as well as have facilities for debugging and error finding purposes. Furthermore, it was desirable that the performance of the application was good enough so that the full potential of the device could be taken advantage of.

5.2.1 Structure

As seen in Figure 5.1, the software developed are sectioned into two parts; one part for the touch algorithm implemented as a library, and one part that implement all the application specific behaviour. The application has two main components; one pointer to a *DeviceInterface* and one pointer to a *ModeInterface*.

Since the access to the device prototype was limited, as well as since extensive use of the device resulted in degrading performance of the touch screen, recordings of the device's input as a user touches it was made at separate occasions. Therefore, since we both wanted to use the device live at some occasions as well as be able to use our recorded touchinputs, an interface was needed so the application could work independently of the input source. This was implemented using C++'s polymorphic capabilities by creating a DeviceInterface base-class that the application calls each frame to acquire a new input-frame.

Furthermore, since different tests were wanted and various properties and capabilities needed to be tested in various ways, a ModeInterface base-class was also implemented. For each frame, the application calls the ModeInterface-member with the previously acquired input-frame and the mode implementation can then do its processing independent of the actual input device or other modes. For example, a test-mode implementation could use the touch library components

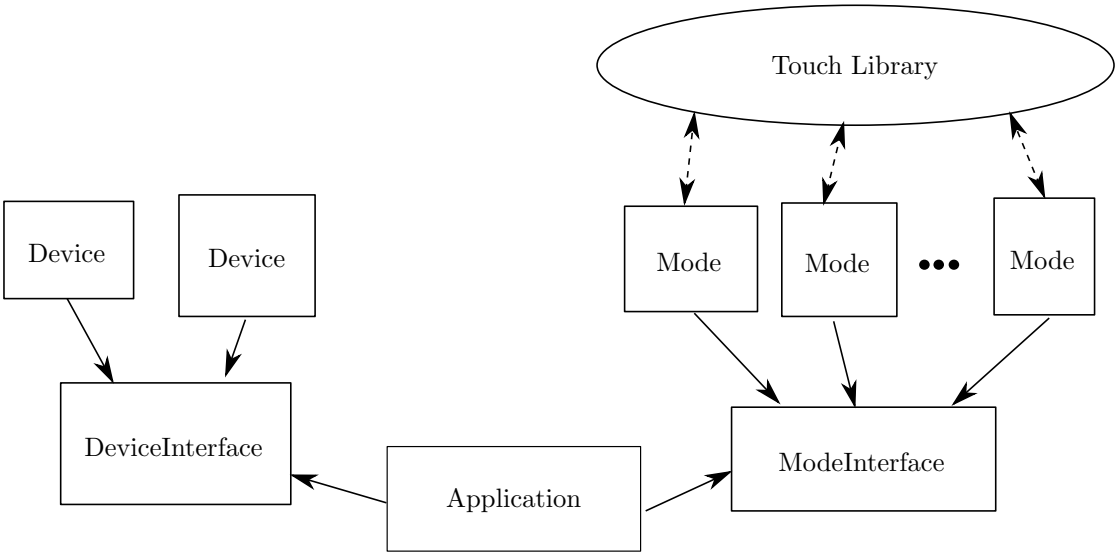


Figure 5.1: Overview of the application structure shown as an UML-diagram.

to test the performance of the algorithm, and another mode could use the input images to acquire data such as pixel values for a specific row of interest and so on.

5.2.2 Performance

As explained in Chapter 3, the device prototype operates in 60 Hz and it was desirable that the algorithm could do its processing in the same time frame. Therefore, some considerations were made in order not to exceed this limit.

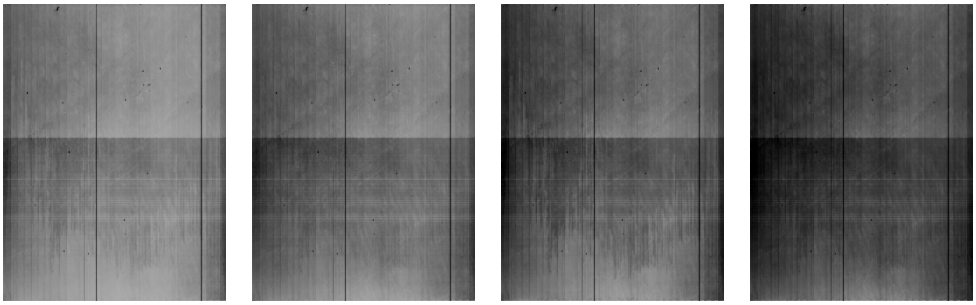
To do this, the free profiling tool *CodeXL* produced by AMD was used in order to search for bottlenecks in the application and algorithm. Using this, inefficient code could quickly be found and dealt with either by rewriting the code for higher efficiency or simply replaced it if the time complexity was too high compared to its importance.

Initially the algorithm used the DirectX library that was previously used in SEL's implementation. However, this library was not capable of efficiently acquiring the currently displayed image, which is necessary for the algorithm as explained in Section 5.5.1, and the graphics backend was rewritten for OpenGL. For this, the open-source multimedia library SFML was included in the project.

5.2.3 Configurable parameters

Since the algorithm utilizes and exposes a great number of parameters that can be fine-tuned, a centralized location that makes it easy to find and change values of the parameters was needed.

For this purpose, a simple system for application settings was made and the basic structure can be seen in Listing 5.1. It supports values for booleans, integers, floats as well as strings and it also uses a single-level namespace system to categorize the options. Each entry also accepts an optional setting for binding the entry to a keyboard shortcut with a value of how much it should be incremented or decremented with. Furthermore, simple variable substitution can be used to substitute values for other variables. An actual example of settings file content can be



(a) Bright Backlight on (b) Bright Backlight off (c) Dark Backlight on (d) Dark Backlight off

Figure 5.2: Examples of the calibration images. Figures (a) and (b) are taken in a bright lighting environment and figures (c) and (d) are taken in a dark lighting environment.

seen in Listing 5.2. It demonstrates the use of shortcut options as well as how the substitution can be used in a convenient way when handling large number of input files.

Listing 5.1 Basic structure of a parameter entry.

```
namespace_name:
    param_name[shortcut settings]: value
```

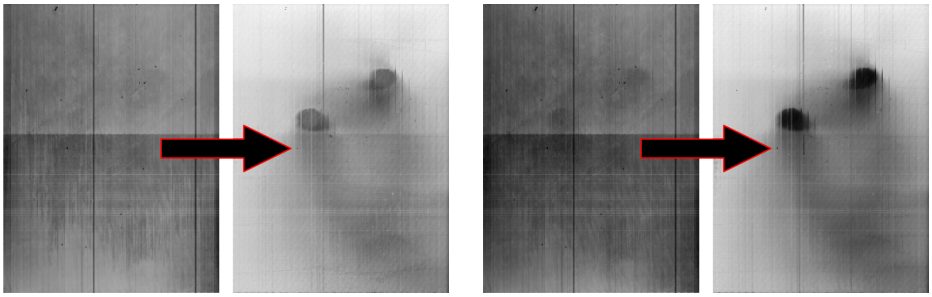
Listing 5.2 A simple example showing usage of the settings system.

```
app:
    threshold[N,5]: 180
    input_file_1: "movie1.avi"
    input_file_2: "movie2.avi"
    input_set: "${app.input_file_1}, ${app.input_file_2}"
```

5.3 Calibration

The first step in the touch detection pipeline after acquiring the input data is to calibrate it using a form of background subtraction. Two calibration images (one each for when backlight is turned on and turned off respectively) are taken beforehand in two light conditions. These light conditions define the extreme points for the operable light conditions, i.e., the calibration images taken in the dark environment is the darkest possible light condition where it is operable and vice versa. Therefore, four calibration images are taken in total; two (backlight turned on and off) taken in dark lighting environment and two (backlight turned on and off) taken in bright lighting environment. An example of these images can be seen in Figure 5.2

Using these four calibration images, the input images acquired from the sensors are calibrated



(a) Backlight on (b) Backlight off

Figure 5.3: Figures (a) and (b) show the calibration transformation of the two input images.

per-pixel as follows:

$$c = \frac{i - d}{b - d} \tag{5.1}$$

where c is the calibrated image, i is the input image, d is the calibration image taken in dark environment and b is the calibration image taken in bright environment. An result after this calibration can be seen in Figure 5.3.

Since the prototype used during the development was slowly degrading as explained in Section 3.2, new calibration images were taken for each usage in order to assure optimum calibration. Furthermore, since it was not possible to turn off the light in the main office, quick and simple capturing of the dark calibration image was made by just covering the device with a diffuse rubber material. However, when correct calibration was of higher importance, the dark calibration image was taken in a completely dark room without covering the device.

Finally, since there is a recurring defect in the sensor image, these are smoothed out to receive a cleaner calibrated image that the image processing will work on.

5.4 Experiment 1: capability examination

After completing the basic structure and functionality of the application, experiments were conducted to evaluate the capabilities of the touchscreen device. The goal of the experiments was to investigate whether touched contact area information would be acquirable under various lighting conditions and if it was good enough to be used for finding touchpoints. This experiment was conducted to investigate what information was obtainable in various lighting conditions, both for black display images and for white display images. The goal of this first test was to get a first rough understanding on what data can be obtained from the device and how it looked.

5.4.1 Experiment setup and execution

A room without direct outdoor windows was used as the place of experiment. The door had, however, a window pane in which light from a corridor could enter. This was neglected in the experiment. The room was equipped with standard office fluorescent lighting mounted in the ceiling. A dimmer was used to change the illumination hitting the touchscreen, and a lux-meter was used to measure the illumination next to the touchscreen. Two sensor images were captured – one where the display image was black and one where it was white – for all the different illumination values shown in Table 5.1.

Experiment 1 illumination values (lux)						
0.1	116	242	602	954	1346	1550

Table 5.1: Measured illumination values (lux) used in the first experiment.

5.4.2 Analysis

The captured sensor data were analyzed and compared with calibration images captured without user interference in the minimum and maximum of the illumination levels respectively (as in Section 5.3). The results are illustrated in Figure 5.4 as curves of separate rows of the sensor data, together with a figure showing the calibrated result (of which the contrast of Figure 5.4e and Figure 5.4f is highly exaggerated to emphasize the touched areas).

From this experiment it was concluded that enough information was available to distinguish touched areas in all the extreme conditions: white display in bright environment, white display in dark environment, black display in bright environment, and black display in dark environment.

However, in very dim lighting conditions when showing a black screen, touch contact areas were not always visible. This would occur, more specifically, when the ambient light falling on the touchscreen has the same illumination as 1) the backlight self-reflected on the glass layer, or 2) the backlight reflected on the fingers. With the exception of this *dead zone*, the experiment result indicated that it would be possible to obtain touchpoint data in varying ambient illumination with different display images, given a new algorithm.

5.5 Proposed algorithm

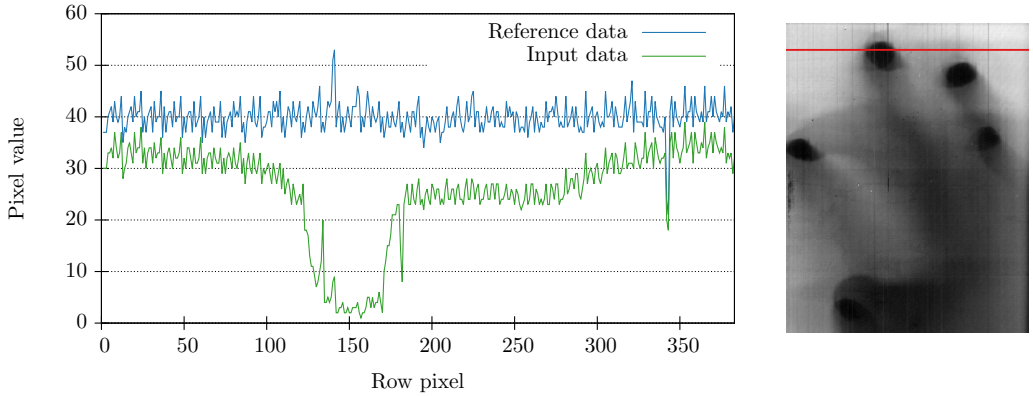
From the data acquired in Section 5.4, a method for combining the information that existed in the input data in the various conditions was proposed. Since the way of processing the data as well as its quality was heavily dependant on the environment lighting as well as the actual image displayed on the device, it was deduced that these two parameters would have to be considered by the algorithm. Therefore, the algorithm investigated should use a method for estimating the illumination as well as the information on what is displayed on the device. With this information available it should process the input data according to the cases demonstrated in Figure 5.4.

This section describes the final algorithm that was developed for the device and later evaluated. An overview of the algorithm is provided in Figure 5.5.

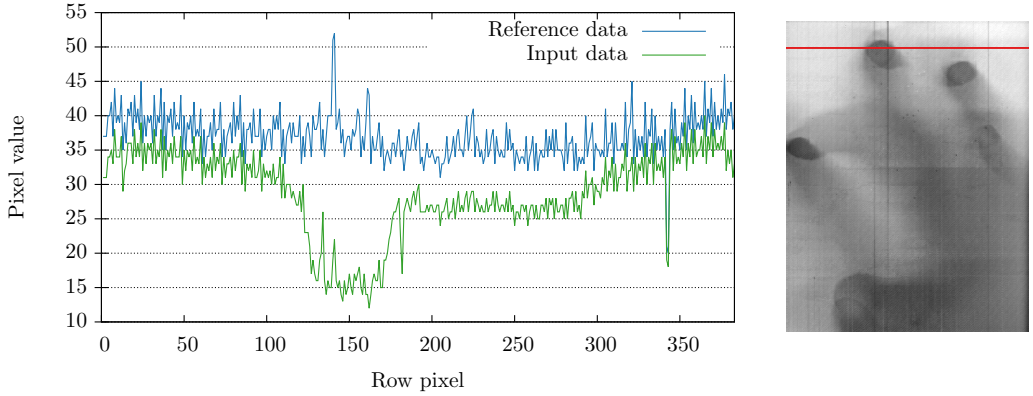
5.5.1 Display image acquisition

The image that is showed on the device's display for each frame (hereinafter referred to as the *display image*) is crucial information for the algorithm. Unfortunately it is quite a time consuming procedure to acquire it. A method that is simple and quick to implement was desired since the task of acquiring the display image quickly is outside the scope of this thesis. One way of solving this relatively easy is by using Windows graphics device interface (GDI), however resources reported that using this API can be a time consuming process. Instead, it was decided to read the back buffer directly from OpenGL since an OpenGL context was provided by the graphics library SFML used in the application. Implementing this is relatively straightforward and it is reported that downloading the back buffer from the graphics device can be done in reasonable time.

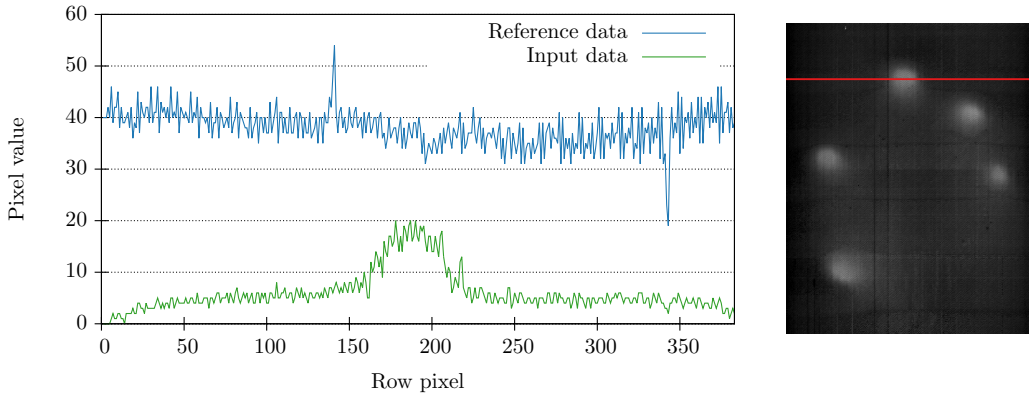
Further challenges was introduced by the fact that the workstation used for development did not have a dedicated graphics card and instead used an integrated graphics unit in the CPU to



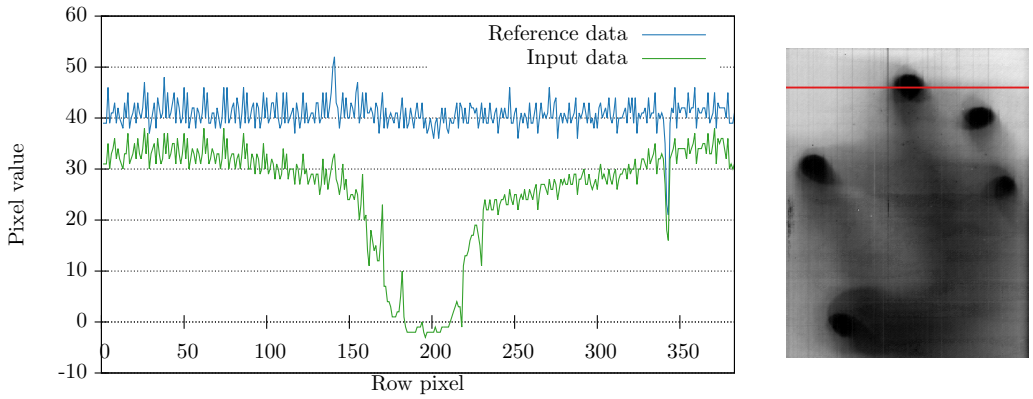
(a) White display image in bright environment (backlight off)



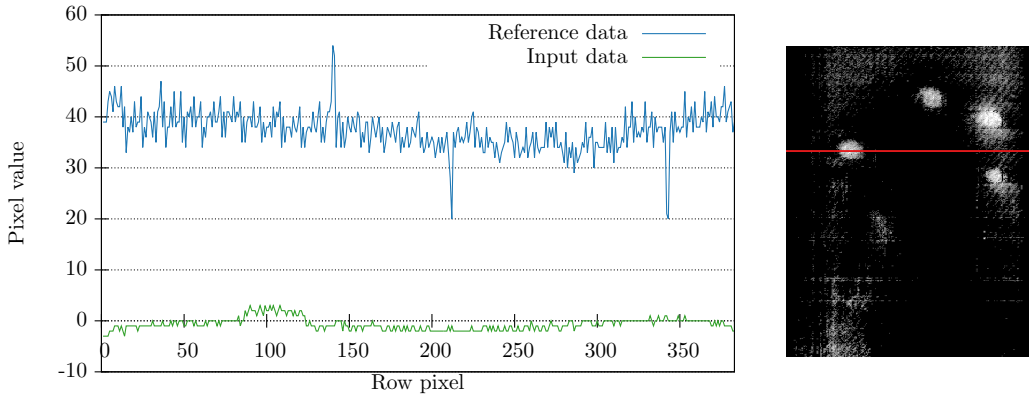
(b) White display image in bright environment (backlight on)



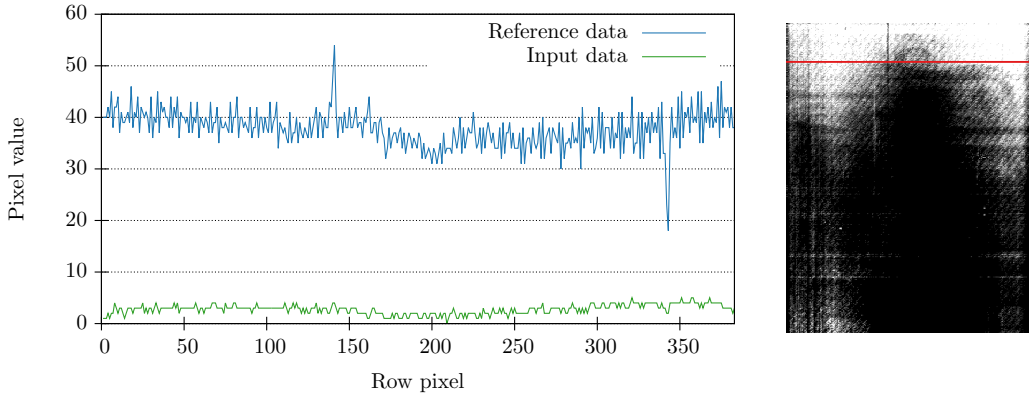
(c) White display image in dark environment (backlight on)



(d) Black display image in bright environment (backlight off).



(e) Black display image in dark environment (backlight on). Although not exposed as clearly in the graph as the previous ones, the data where the touch area is located differs about one or two values compared with the dark calibration image. Contrast highly exaggerated to emphasize the data.



(f) Black display image in dark-dim environment (backlight on). Contrast highly exaggerated to emphasize the data. Nothing can be extinguished, but the big black mess of a shadow the hand is casting.

Figure 5.4: Left column showing a single row of data compared with its reference images (bright and dark ambient lighting). Right column showing the image after the calibration process where the red line is the sampled line in left column.

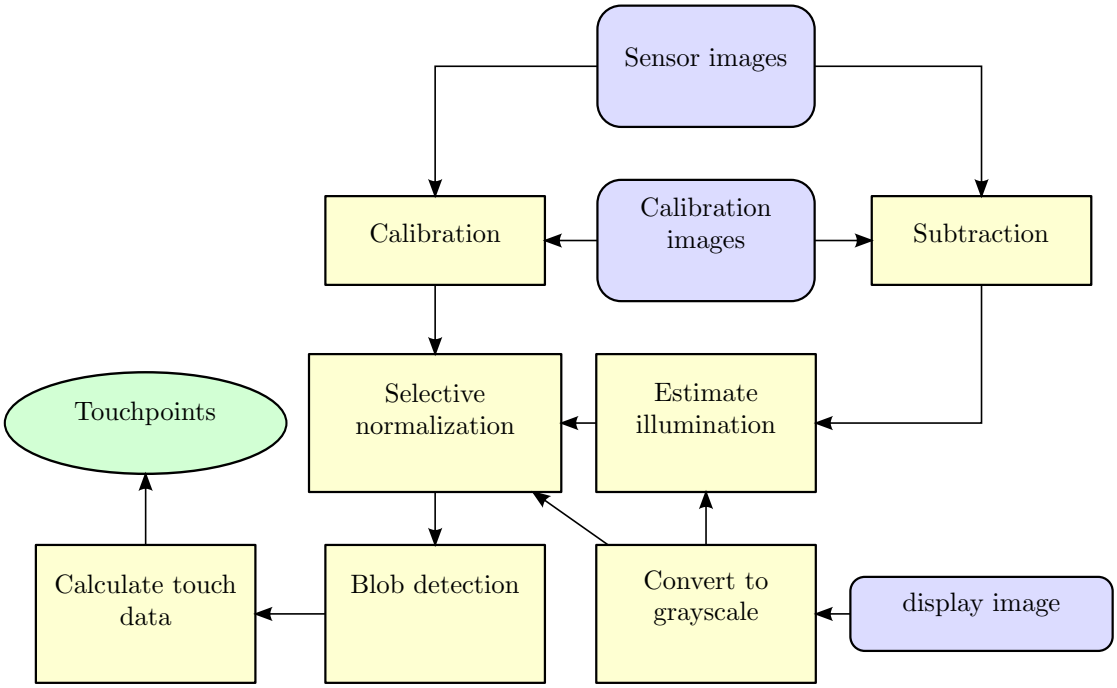


Figure 5.5: An overview of the algorithm shown as a flowchart.

handle graphic procedures. See section Section 5.1 for description of the hardware. Compared to the workstation hooked up to the device which could acquire the back buffer in almost real-time, the workstations for development took almost half a second to perform this task, which is too slow even for a prototype. In order to reach reasonable times, the back buffer was required to be downscaled, a procedure that can be done on the graphics unit, to a size of 48 x 64. Since the new width and height are 8 times smaller than the sensor resolution, the precision is greatly reduced and finer details of the display image are neglected.

Discussion

Finally, in an embedded system, this would not be much of a problem since all data exist in the hardware. By integrating the algorithm in the hardware, it would be possible to have direct access to the image shown on the display. Therefore, even if the current method has much smaller resolution, or very high latency, it is not a future problem.

5.5.2 Illumination estimation

The surrounding ambient illumination can be estimated using the acquired optical sensor data. In order to work on data with correct offsets, the sensor data is subtracted with a calibration image previously captured in the darkest allowable condition showing a black display image. Only the data captured with the backlight turned off is used in order to avoid self-illumination inflicted by the backlight.

To filter out high frequency noise and erroneous sensor values if any, the subtracted image is smoothed using a filter that averages the neighboring pixels. Whilst doing this, the image

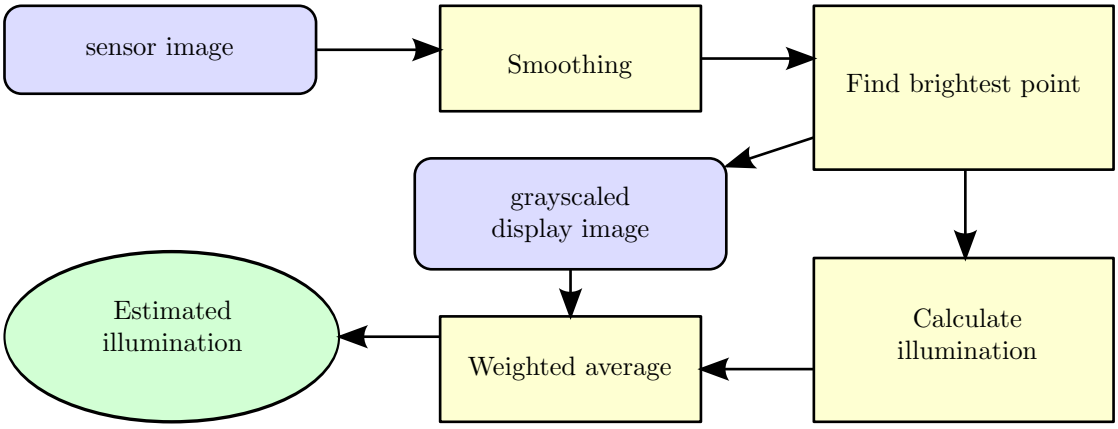


Figure 5.6: Flowchart showing the steps for estimating the ambient illumination

can be downscaled to save processing time when finding the brightest point. A linear search is performed over the pixels, returning both the maximum pixel value and its 2D coordinate.

By using pre-measured data mapping sensor values into lux, the illumination can be estimated. For obtaining as good estimation as possible, not only one, but two functions were fitted – one each for black and white display images. The previously found maximum pixel value is then fed into these two functions separately, resulting in two illumination estimations, assuming its corresponding coordinate in the display image was completely black or completely white. Finally, the brightest point's corresponding 2D position in the grayscaled display image is used as a weight to calculate the average of the two mapped illumination values.

Furthermore, to stabilize the estimated illumination value and keep it from momentarily dipping as the user interacts with the display, a history of past estimated illumination values are stored, and the largest of them is used as the actual return value.

In our implementation the history was set to contain the five most-recent values, and the offset corrected image was shrunk to match the processed display image. Despite making the image smaller, the linear search over the pixels still ended up taking a lot of processing time. However, by calculating the illumination value at half-second intervals, the processing time was made negligible.

Discussion

By introducing a delay (i.e., only calculating new illumination estimation after certain time intervals), the touch acquisition algorithm as a whole will be less responsive to sudden changes in the surrounding lighting environment.

5.5.3 Selective normalization

The main step in this algorithm is the combination of all the data discovered in the first experiment described in Section 5.4. This part of the algorithm tries to process the sensor images to produce an image with clear touchpoints in a wide range of conditions. It is done by using all the input data explained previously, i.e., the display image described in Section 5.5.1 and the estimated illumination described in Section 5.5.2.

The main idea is that the amount of transmitted light through the LCD layer is related to the

color of the LCD element and since and this value is one-dimensional, a transformation from a RGB value to a single value is possible. Since the black color is the least transmissive and white color is the most transmissive, it is deducted that the transformation can be approximated with a luminance-preserving color-to-grayscale transformation. During the development process, the built-in color-to-grayscale transformation found in OpenCV was used.

Next, the image processing is divided into three cases; where the display image is considered white and one where the display image is considered black for dark and bright light conditions respectively.

White display image processing

During the experiment it was learned that the difference between the two sensor images reliably indicates the touchpoints in all lighting environments. With this knowledge, the image representing a white display image, wh, is given by:

$$wh = (b_{on} - b_{off}) \cdot k \tag{5.2}$$

where b_{on} and b_{off} is the calibrated input images with backlight turned on and off respectively, and k is a constant introduced to scale the result such that white represents no touch and black represents full touch. Since the amount of transmitted light is considered to be independent of the ambient illumination, k can be constant and estimated beforehand. This is done to simplify the actual touchpoint detection described in process Section 5.5.4 that would otherwise have to take the display image into consideration.

Black display image processing

In the case for black images, the illumination has to be considered. In bright light conditions, the shadow information can be used to detect touchpoints. However, in dark light conditions, there is no light source that can cast a shadow onto the touch device. Although, since the first experiment described in Section 5.4 showed that there existed data in the sensor image with backlight turned on, albeit with very low dynamic range, this can be used for detecting touchpoints in these conditions. Thus, the input to use for black display images, bl, are chosen according to:

$$bl = \begin{cases} b_{on} \cdot k & \text{if } ill < \gamma \\ 1 - n(b_{off}) & \text{otherwise} \end{cases} \tag{5.3}$$

where k, b_{on} and b_{off} are the same as in Equation (5.2), γ is a predefined threshold value inside the dead zone (explained in Section 5.4.2) and ill is the estimated illumination calculated in Section 5.5.2. The function $n(x)$ serves the same purpose as the linear scaling in Equation (5.2), that is to remap the values from white which represents no touch to black which represents full touch. However, in difference to the process white display images, the scaling is not constant and is in fact dependant on the light conditions. To solve this, the function $n(x)$ is implemented as a simple histogram normalization which stretches the brightest graylevel in the result to white. This process is explained in Algorithm 2.

Combination

Finally, by knowing the transmittance of the display, the amount of wh and bl needed to be mixed together can be chosen. The function for doing this procedure was chosen to be a binary function, since taking the difference is more reliable and independent on the lighting conditions

Algorithm 2 Algorithm for rescaling gray colors in the image to white.

```
1: function HistorgramStretch(image, limit)
2:     hist[256] ← make_histogram(image)
3:     max_value ← 255
4:     non_empty_bins ← 0
5:     bin ← 255
6:     while bin ≥ 0 do
7:         if hist[bin] ≠ 0 then
8:             if non_empty_bins = 0 then
9:                 max_value ← bin
10:            end if
11:            non_empty_bins ← non_empty_bins + 1
12:        else
13:            non_empty_bins ← 0
14:        end if
15:        if non_empty_bins = limit then
16:            break
17:        end if
18:        bin ← bin − 1
19:    end while
20:    if non_empty_bins ≠ limit then
21:        max_value ← 255
22:    end if
23:    RescaleValueToWhite(image, max_value)
24: end function
```

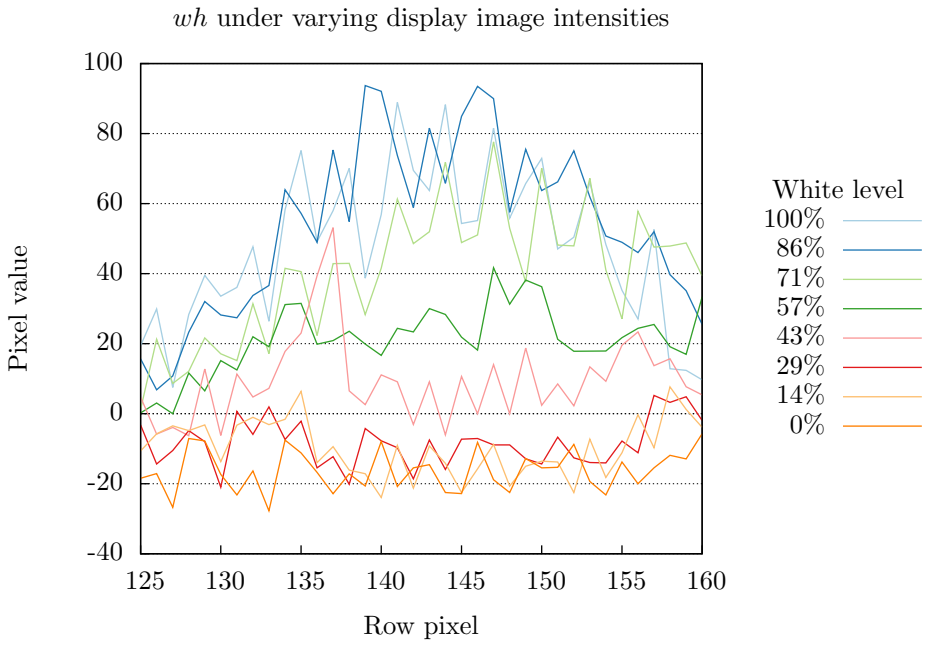
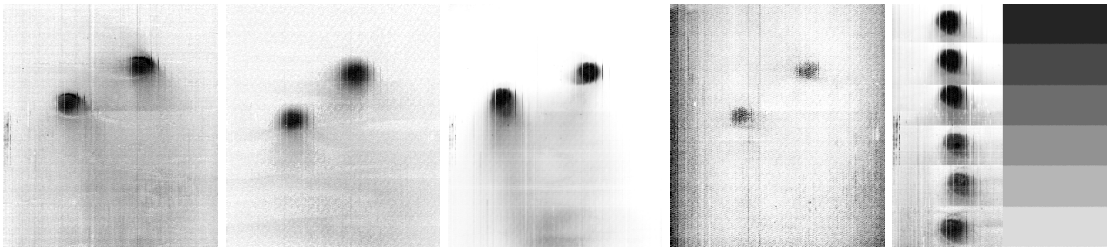


Figure 5.7: Showing how wh (i.e., the difference between b_{on} and b_{off}) is affected by the intensity of the displayed image. The different curves are ranging from a complete black display image (0.0) to a complete white one (1.0). Notice how only the most upper three curves clearly peak at around row pixel 140 to 145, thus indicating a touched area.



(a) white bright (b) white dark (c) black bright (d) black dark (e) graylevels

Figure 5.8: Showing the result for four various light conditions and display images as well as an image with varying graylevels.

(illumination levels and directions) compared to using only the sensor image taken with backlight turned off. Using the data shown in Figure 5.7, the threshold for this operation was chosen as 0.71. This is done by the function:

$$I(x) = \begin{cases} 0 & \text{if } x < 0.71 \\ 1 & \text{otherwise} \end{cases} \tag{5.4}$$

where x is the actual value of the display image ranging from 0 to 1. The two images wh and bl are then mixed using:

$$m(wh, bl) = wh \cdot I(i) + bl(1 - I(i)) \tag{5.5}$$

where i is the actual value of the display image. Some examples of the result are shown in Figure 5.8.

5.5.4 Image segmentation and blob extraction

After the steps in Section 5.5.3 have enhanced the touchpoint features as much as possible, the actual touchpoints have to be found.

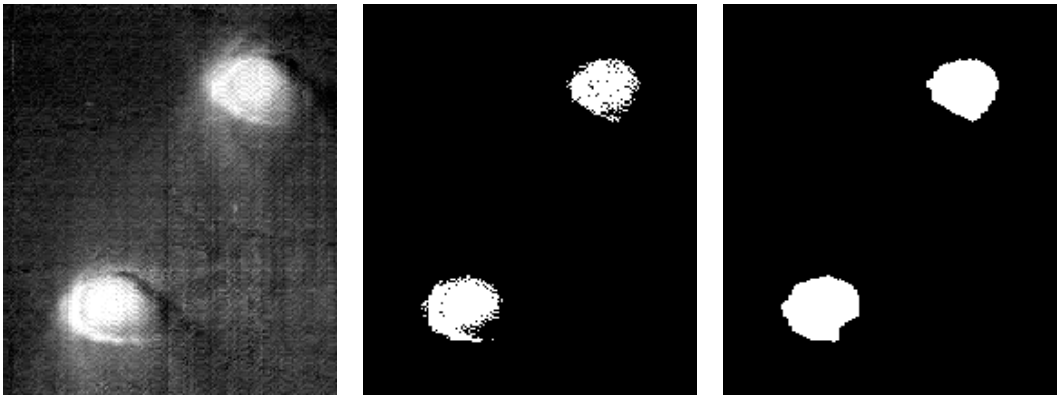
Thresholding

First, the resulting image are made binary using a global threshold which converts the values in the following way:

$$b = \begin{cases} 1 & i > \beta \\ 0 & \text{otherwise} \end{cases} \tag{5.6}$$

where b is the binary image, i the image retrieved from Section 5.5.3 and β is the threshold. β has empirically been chosen to 205. The result after this transformation can be seen in Figure 5.9b.

Secondly, before finding touchpoint regions, the thresholded image is transformed using mathematical morphology operations with an elliptical structuring element; first a close operation and subsequently an open operation is performed. The whole transformation from selective normalization to mathematical morphology operations can be observed in Figure 5.9.



(a) Results after Section 5.5.3

(b) Thresholding the results after Section 5.5.3.

(c) Transformed image after the closening and opening. Notice that holes and peninsulas are reduced.

Figure 5.9: Showing the results after each transformation step.

Connected-component labeling

Finally, all pixels that are connected with each other that form the touchpoint areas needs to be identified. This is done by performing a connected-component labeling described in Section 4.2. In the implementation used, a parameter *kernel size* can be set that decides the search step size in order to increase the search efficiency. Since the size of the touch region is roughly know, this parameter can be chosen to be small enough to still find any touchpoints whilst minimizing the search time. In our implementation, the kernel size is chosen to be 8 pixels. Parameters such as maximum and minimum touch area and contour size, width vs. height ratio, and contour fill ratio were also implemented. See Figure 5.10 for an example of the labeling process result.

5.5.5 Touchpoint calculation

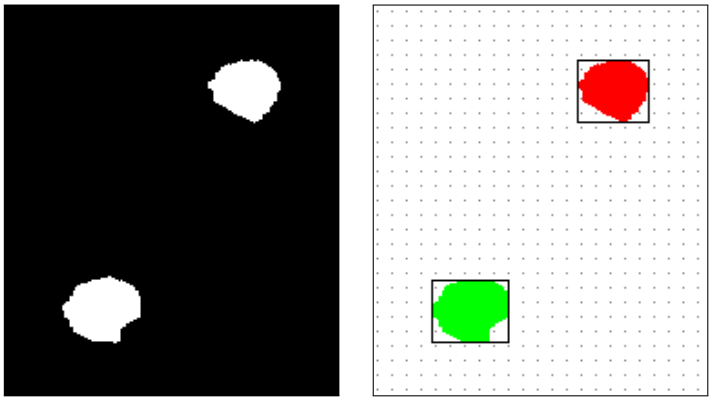
After the touchpoints have been identified and segmented, they are analyzed and their touchpoint coordinate is being calculated.

Three implementations were made for the calculation of the touchpoints: bounding-box centering, geometric centroid, and weighted centroid. The simplest method, box centroid, simply searches for the extreme positions of a touchpoint area and creates a bounding box from the minimum and maximum values respectively. After this is found, the position is calculated by simply taking the center of the box.

The latter two methods uses the actual geometric shape of the touch area. The touch area's bounding box is used to crop the mathematical morphology improved thresholded image in order to obtain only the shape of that touch area. Then a centroid of that shape is calculated, either with or without weights (i.e., the pixels' values).

Discussion

Since the touched contact area returned by the selective normalization often tends to be circular shaped, and its graylevel intensities distributed evenly, the above three mentioned center methods will end up calculating nearly the same center. This is illustrated in Figure 5.11. Because of



(a) (b)

Figure 5.10: Figure (b) shows the result after the connected-component labeling.

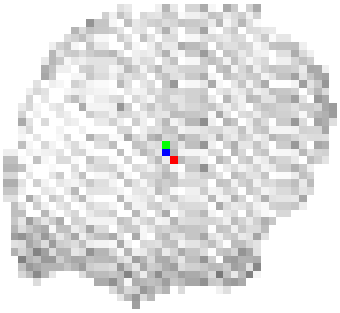


Figure 5.11: Showing the three different implementation of center calculation. Red pixel indicates the center calculated by the bounding-box method, whereas the green is from weighted centroid, and blue is from geometric centroid. Notice how little they differ.

this, the best suited method when taking processing time into consideration is to use the simple bounding-box method.

5.6 Experiment 2: algorithm evaluation

The purpose of the second experiment was to reevaluate experiment one with more accuracy and getting a better estimation of how the optical sensor values are mapped to the actual illumination falling upon them. The experiment was executed with more suitable equipment in a larger number of illumination conditions.

5.6.1 Experiment setup and execution

For the second experiment the same room without windows was chosen as the place of experiment. However, this time pieces of cardboard were cut out and fitted to the windows in the door, blocking light from entering. A fiber-optic metal-halide light source, Moritex MME-250, with an intensity of illuminaton up to 270,000 lux, was used to illuminate the scene metal halide light source [2014]. Since the touchscreen does not contain any optics (see Section 3.1), light can enter

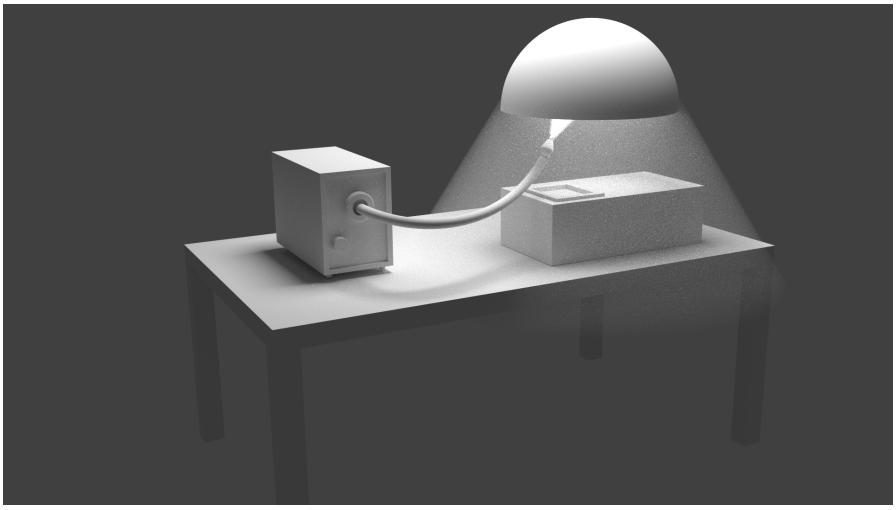


Figure 5.12: Illustrative overview of the diffuse ambient lighting model experiment setup.

from any direction making the sensors capture a diffuse image of its surrounding. In order to simulate an environment with several different light sources a diffuser was built using plain fiber document paper and tape. It was placed directly above the device, hanging from a make-shift built contraption. The fiber-optic light source was places and directed such that the directional light rays would scatter in all directions, effectively simulating *Lambertian reflectence*. A lux-meter was placed next to the touchscreen and was used to measure the incoming diffuse light. See Figure 5.12 for a illustrative overview of the setup.

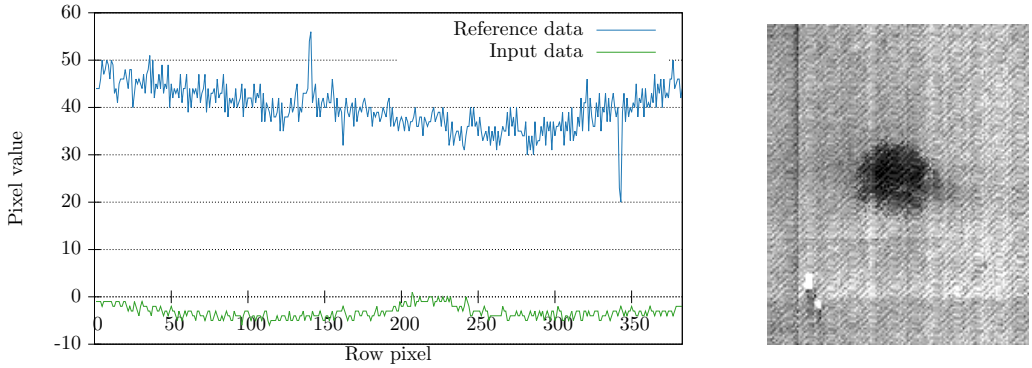
Two sensor images were captured in the same way as in experiment one – one where the display image was black and one where it was white – for the 21 different values shown in Table 5.2.

Experiment 2 illumination values (lux)						
0.35	1.3	6	21	46	79	111
120	148	152	204	207	312	490
723	1067	1283	1405	1615	1820	2000

Table 5.2: Measured illumination values (lux) used in the second experiment

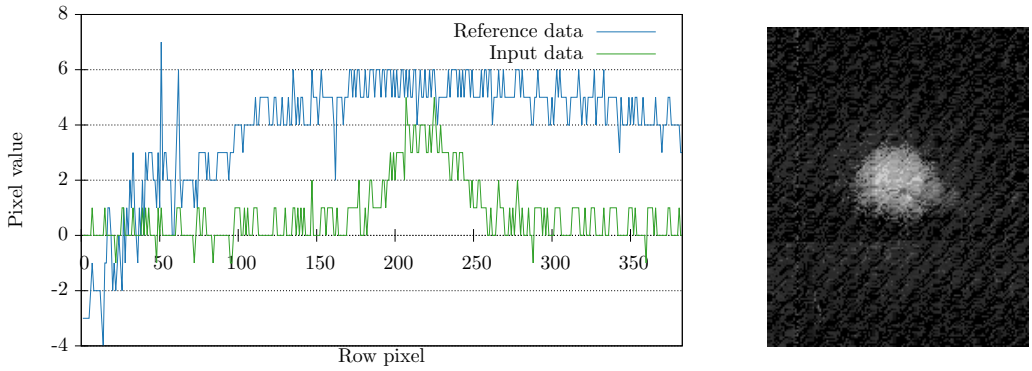
5.6.2 Analysis

After capturing all the sensor images, they were analyzed to see what information was contained in all the different lighting conditions. It was discovered that the result differed from the first experiment when showing black display images. See Figure 5.13 and compare with Figure 5.4e. Contained in the code previously developed by SEL, a code snippet regarding a calibration scheme using four sensor data images (dark and bright environments showing a black and a white image respectively) were found. However, this snipped was never used in the code. By implementing this four way calibration scheme – calibrating the input sensor data (backlight turned on and off respectively) both with black and white captured calibration images – a total of four calibrated images were used as input to proposed algorithm.



(a) (b)

Figure 5.13: Black display image in dark environment (backlight on). Here, during experiment two, the input value is lower than the dark reference data, thus inverting the intensity levels.



(a) (b)

Figure 5.14: Same input as Figure 5.13 but now it is calibrated using *black* calibration images rather than white ones.

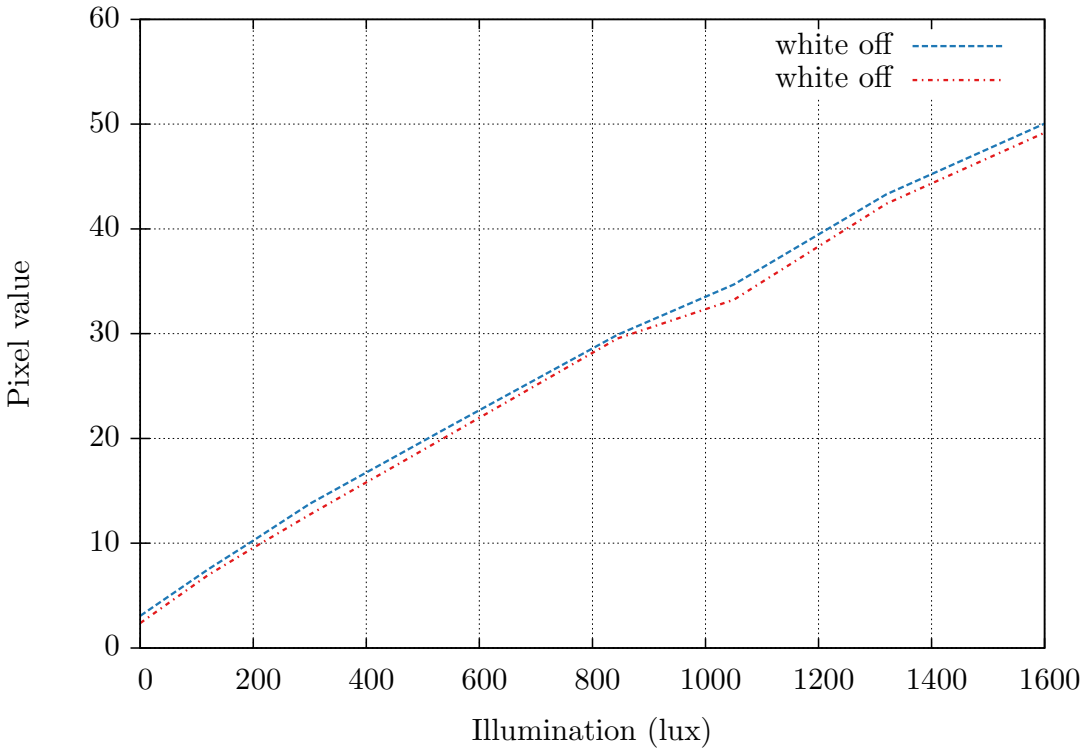


Figure 5.15: Showing two curves that were used to fit pixels to illumination values.

Finally, as was the main purpose of this experiment, the sensor data which were captured with a black display image were searched for their maximum pixel value. This value was then was mapped to the corresponding measured illumination. Using these mappings from a pixel value to a illumination value, two functions (one for black and white display image respectively) were fitted using quadratic curves. These functions are used in Section 5.5.2 and are plotted in Figure 5.15.

Discussion

Despite the fact that the sensor data shown in Figure 5.13 did not resemble the same condition in experiment one, it is not considered as a calibration error. Following this experiment, several other sensor captures were taken, all with the same result regarding black displayed images. In the first experiment however, black calibration images were not captured and were not considered necessary. This might have been due to incorrect capturing (but with rather good result) of calibration images at that time. Another factor might be that the sensors have been such degraded that they behaved differently.

During this experiment, only illumination values up to 2000 lux were being tested. Fitting a better curve that is also valid for higher illumination values (i.e., outdoor environment and direct sunlight), might also be desirable. Nevertheless, since the touchscreen was a prototype and not fully tested, too high illumination might lead to sensor damage, and it was deemed out of scope for this thesis.

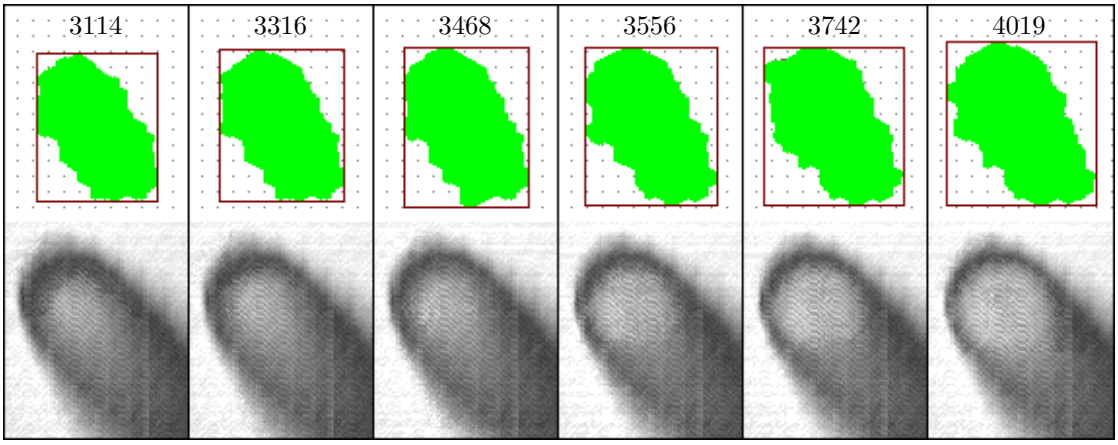


Figure 5.16: This image shows the labelled touch area and the calibrated input image with backlight turned on for six different frames while a user gradually presses harder on the device. The number of pixels contained in the touch area is denoted on the top of each frame. Note how this number increases for each frame as the user presses harder and harder.

5.7 Experiment 3: Interaction methods

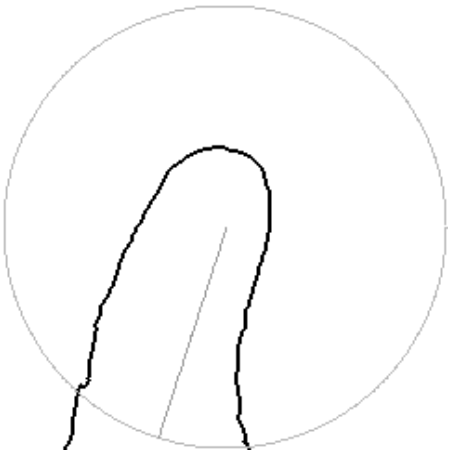
In this section, an experiment about what touch features can be extracted is executed. After that, a way to obtain pressure and finger rotation information is described. Finally, this section will focus on multi-touch gestures and how to track individual touchpoints in time.

5.7.1 Finger orientation properties

The goal of this experiment was to investigate what kind of orientation information could be obtained. The tested properties were finger pressure, finger rotation, finger incident angle, and whether the distance a finger is from the screen could be measured. The experiment was performed in standard office lighting condition by capturing and recording sensor images and videos of fingers on top of the device. For finger pressure, a video of a finger touching a fixed point with increasing pressure was recorded. For rotation, a video of a finger rotating one revolution around a fixed point was recorded. For incident angles, a finger were captured pivoted at 15, 25, 45, 65, and 90 degrees from the display plane. And finally, for finger hovering, sensor image data were captured with a finger from touching state and increasing 2 mm for every image up to 20 mm.

Pressure

Since the pressure generally manifest itself by generating a larger touch area as can be seen in Figure 5.16, this is used for messuring the touch pressure. This method is a simple but common way for calculating pressure for optical touch devices. In the labeling process, the number of pixels belonging to a single touch point are counted as each pixel get processed. When finished, the value is sent as it is together with all the rest of the touch data to the host application.



(a) Debugging data of the rotation calculation. The thick black outline is the edge detected finger. The gray circle is the searched perimeter and the gray line stretching from the center is the calculated rotation.

(b) One of the sensor images (backlight turned off calibrated as described in Section 5.3) from which the rotation is calculated.

Figure 5.17: The result of rotation calculation.

Rotation

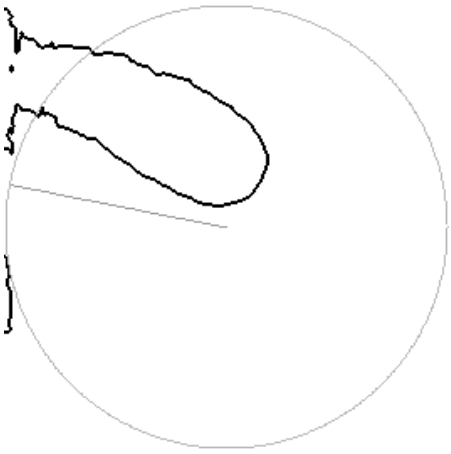
As a proof of concept, a simple procedure for calculating the finger rotation was implemented. First, the image was cropped to the local region around the touch area. Secondly, a simple edge detection algorithm based on mathematical morphology was executed on the binary thresholded image (as descriped in Section 4.1). This is implemented by taking the difference between the eroded and diluted image. After this, the procedure searches the perimeter of the circle lying on the touch position with a specified radius. After two edges have been found on the perimeter that are of satisfying distance to each other, the angle between these to points are returned as the finger rotation. This process is illustrated in Figure 5.17.

However, as mentioned, this was only a simple proof-of-concept implementation that is not very stable and there are many corner cases where it does not work. Two examples of such cases are showed in Figure 5.18.

Discussion

Both of the implementations for the pressure and finger rotation calculations were only proof-of-concept procedures to examine what further data could be extracted. Although some results where produced, they were in general not very reliably and further work would be needed to produce more robust and accurate calculations. However, since we believe it is possible, we think it would be worth investigating these properties by taking the actual gray levels of the input images into account, not only the shapes.

For example, the pressure could be calculated by examining the fall-off from the touch position down to the edge of the finger. As can be seen in Figure 5.16, lower pressure is indicated by a slow and smooth fall-off from the touch center, where as for high pressure is indicated by sudden change of graylevels very close to the border of the finger.



(a) An example of where wrong results are calculated. In this case, there is noise in the edge of the image that is found on the searched perimeter that results in a faulty angle.

(b) An example when the algorithm fails to find the angle. In this case, the hand is close to the surface and generates an ill-shaped finger that has a too wide angle at the given radius. In this scenario, a smaller radius are preferred.

Figure 5.18: The images shows examples of where the rotation calculation fails on providing accurate results.

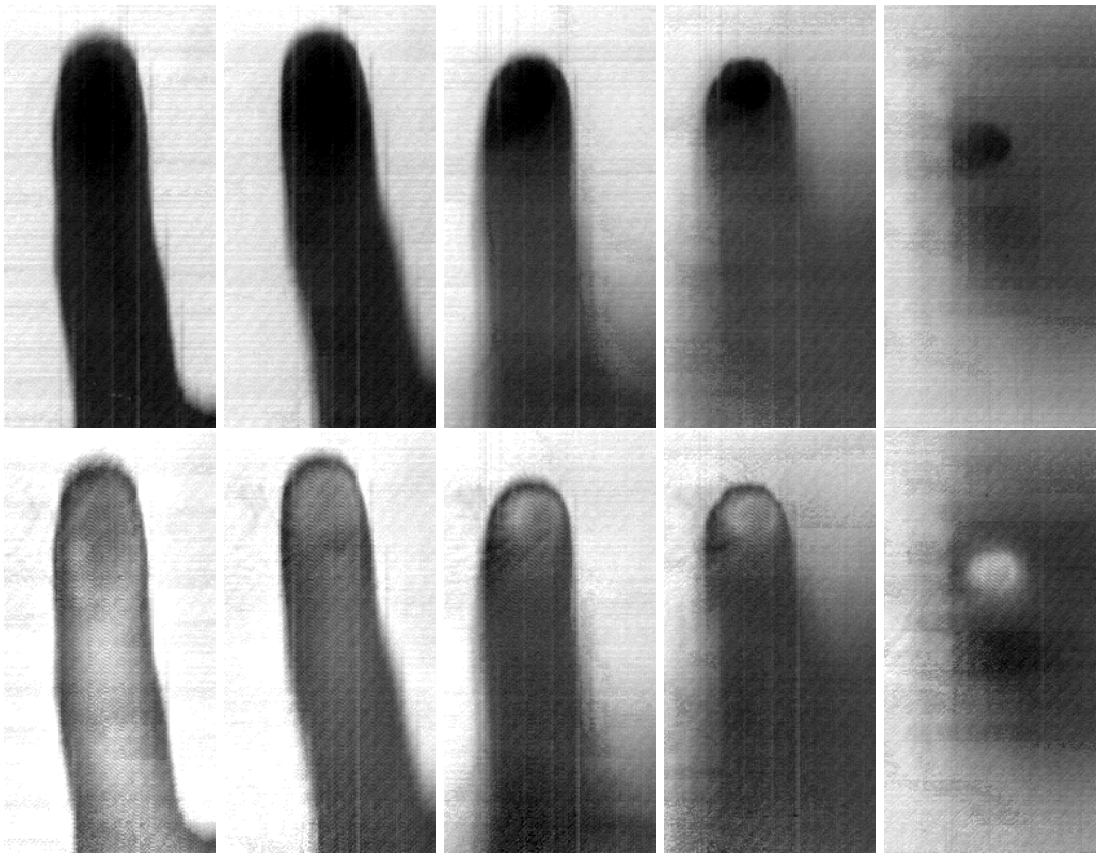
Similar technique could be used to calculating the rotation of the finger. The rotation could be given by finding the direction where the slowest fall-off occurs from the fingertip.

Furthermore, due to lack of time, incident angle and hovering properties were not fully investigated nor implemented. Some of the captured data can be seen in Figure 5.19 and Figure 5.20. From this, we believe that the same graylevel intensity fall-off technique as mentioned above could be used in order to calculate the finger's incident angle. However, when it comes to obtaining the finger's height hovering above the screen, the captured data get diffused quickly. Already at around 4 millimeters above the screen the coordinate of the fingertip can not trivially be located, especially under more unfavorable lighting conditions. Even though obtaining directional movement hover information is a simple task (e.g., swiping a hand over the screen), acquiring 3D spacial information would require a more advanced approach, such as analyzing shapes and the diffuse light spread.

Finally, the pressure and rotation properties are vary dependant on the ambient illumination. As can be seen in Figure 5.4e, there are very little information given in dark illumination with a black display image and the information given is very noisy and of very low dynamic range (with only a few integer values in difference). Because of this, it might be very difficult or even impossible to acquire information such as pressure and rotation in these conditions. This would also need further examination.

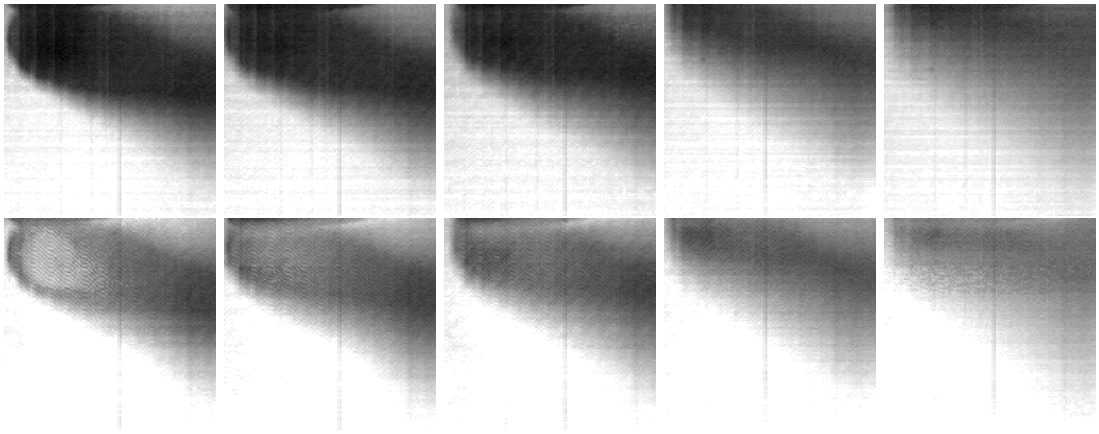
5.7.2 Tracking

The last block in the pipeline is the tracking algorithm for the touchpoints. In all previous steps, the history is not taken into account and consideration is not made to preserve the identity of the touchpoints between frames. However, this is important since a touch is a temporal action and care needs to be taken in order not to confuse multiple touchpoints between each frame.



(a) 15 degrees (b) 25 degrees (c) 45 degrees (d) 65 degrees (e) 90 degrees

Figure 5.19: Comparison of the incident angle for a finger touching a fixed point. The data were captured in a bright environment showing a white image. The upper row shows the calibrated input when the backlight is turned off, whereas the lower row shows it when the backlight is turned on.



(a) Touching (b) 2 mm above (c) 4 mm above (d) 12 mm above (e) 20 mm above

Figure 5.20: Comparison of a hovering finger's distance to the screen. The data were captured in a bright environment showing a white image. The upper row shows the calibrated input when the backlight is turned off, whereas the lower row shows it when the backlight is turned on.

This is especially important for calculating touch-states (such as going down, touching, going up etc.) as well as for using multi-touch gestures.

Since the tracking needs to take the previous results in consideration, further improvements can be done to enhance the user experience. The stability of a touch and drag was increased in this step as it could keep track of touchpoints that is lost only for a short time (e.g. a few frames) without signaling it as non-touching. Only basic functionality was wanted and thus time was not spent on finding and implementing a more sophisticated algorithm.

Implementation

The general idea in the tracking algorithm is to store a history of the touchpoints from the previous frame and do a comparison of the current touchpoints. If a touchpoint in the history lies within a fixed radius to the current touchpoint, it candidates to be chosen as the previous touchpoint and added to a list of touchpoints being close enough. When all previous touchpoints have been considered, the touchpoint being closest are chosen as the previous touchpoint and its identification number is assigned to the current touchpoint. If there is no previous touchpoint being close enough to the touchpoint or if all previous touchpoints in its list are already assigned to another touchpoint, it is being assigned a new unique touchpoint identification number.

After all the current touchpoints have been assigned an identification number, they are processed for determination of their state. If the touchpoint's identification number does not exist in the history, it is considered a new touchpoint and is therefore going down. In the case where the touchpoint's identification number did exist in the history, it is considered as currently touching on the touchscreen. However, if there is a touchpoint identification number in the history that does not exist in the current frame, it is flagged as going up and reinserted in the current touchpoints. Doing this, touchpoints that, for some reason, are lost in one frame's time can be recovered if they exist in the following frame.

Discussion

This is a naive implementation that is only keeping track of one previous frame. A further enhancement of the algorithm would be to increase the number of tracked frames in the history. However, this will introduce a delay but since the display is operating in 60 Hz, adding a few extra frames to the history will not make a noticeable difference.

Furthermore, with a larger history, it would also be possible to predict the movement of a touchpoint. By doing this, if there are several touchpoints close enough to a touchpoint, it would help to choose which one should be chosen as its predecessor by choosing the one best fitting the prediction of the touchpoint.

Finally, the predecessors are chosen as they come in the list they are stored. It would be better to consider the whole picture when choosing the predecessor.

5.7.3 Multi-touch gestures

With a functional tracking system that enables touchpoints to keep their registered identification number in both space and time, a natural extension was the addition of multi-touch gestures. Gesture recognizing is a wide area within image processing and machine learning and can be made into a thesis by itself. There is also already a great number of existing implementations and library collections for this purpose. Due to the limitations of this thesis, it was decided to not implement one from scratch. Likewise, demo applications that make use of multi-touch gestures were also deemed out of scope for this thesis.

Microsoft is providing a so called Touch Injection API for its operating system Windows 8 [Microsoft Corp., 2014]. It can be used for simulating touch input into the Windows environment. Since our application was already being developed using Microsoft Visual Studio and SEL had workstations that run Windows 8, the Touch Injection API seemed like a good candidate.

Moreover, since the release of Windows 8, Microsoft have promoted its new multi-touch gesture based desktop environment know as Metro. This has resulted in a lot of applications utilizing this technology in the Microsoft Store.

Implementation

Since our application already acquires all information that is necessary for the Touch Injection API, the Implementation was straightforward. For all obtained touchpoints in every loop of the application, data such as position, touch state, bounding box size etc., are being pushed into the API. The internal algorithms of the API then interprets the list of touchpoints and its history to simulate multi-touch gestures, which are directly sent to the Windows environment as input.

Chapter 6

Empirical evaluation

Using the DECIDE-framework as guidelines explained in Section 4.3.2, the evaluation was planned in the following way.

The main purpose of the evaluation was to do a formal evaluation of the algorithm proposed for the touchscreen device. Further goals were also to do an evaluation of the accuracy as well as the robustness of the system. It was also of interest to get an understanding whether or not the system behave intuitively to the user, i.e., test whether it meets the expectations of a modern user with previous touchscreen experiences.

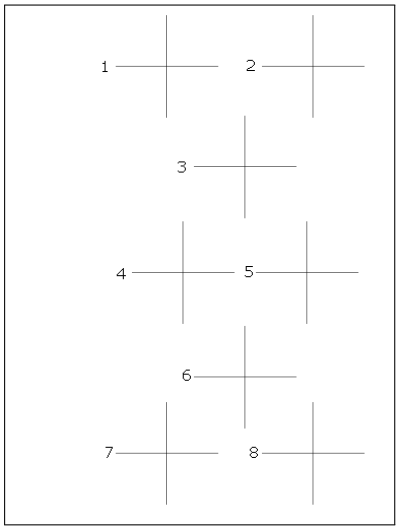
The main question in the study was "How well does the system perform?". This can be subdivided into further sub-questions; "How well does the system perform in various lighting conditions?" and also "How well does the system perform with various display outputs?" (cf., black display problem in Section 3.2). Finally, we also want to answer the question "Is the system intuitive and easy to use?".

For the evaluation, quantitative results were wanted, since it is easy to visualize and present in the report. Because of this, *Usability testing*, explained in Section 4.3.3, was deemed suitable. The test was designed so the time it took to complete the test could be measured as well as the number of errors. A questionnaire was also designed as a qualitative evaluation to give feedback on more subtle aspects that are not clearly reflected by the metrics only.

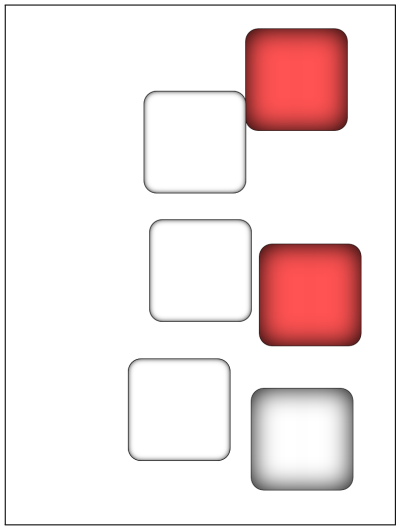
A number of practical issues also needed to be dealt with. Since almost no one of our coworkers could speak English, a way to deal with the language barrier present was needed. Furthermore, it was not possible to use test subjects from outside the company, and because of that, our coworkers would have to put their own time from work and spend it on our study. Therefore, the supervisors would have to make sure that this is possible. Other practical issues with the test was how all the collected data would be stored. Since all test data collected from a single subject would be around 6 GB and the computer connected to the device had limited storage, this problem had to be considered. Finally, since we wanted to do the test with different parameters (mainly different illumination and display output values) human factors play a major role in the test. Care need to be taken not to commit such organizational errors.

The study should be completely anonymous and this should also be clear for the subjects. Video would also have to be recorded to analyse the way the user operates the device. Care need also be taken here not to invade on the subject's privacy, thus only the relevant information should be recorded.

Finally, the data should first of all be presented quantitatively as an easy and quick way to show the results. However, the questionnaires should be analyzed and discussed as well for deeper understanding of the system and how the subjects experienced it.



(a) One of the crosshair images that was displayed on the screen during the evaluation. Two more images were also used, one with black background and one with gray background (both showing white crosshairs).



(b) Button pressing mode. The three states of the button were: inactive, currently pressed, and activated, represented by no fill, gray inner shadow, and red fill respectively.

Figure 6.1: Crosshair and button targets for empirical evaluation.

6.1 Test outline

The test was executed with three different display images as well as three different lighting conditions resulting in nine different test states:

1. White display image in bright environment.
2. Gray display image in bright environment.
3. Black display image in bright environment.
4. White display image in dim environment.
5. Gray display image in dim environment.
6. Black display image in dim environment.
7. White display image in dark environment.
8. Gray display image in dark environment.
9. Black display image in dark environment.

where bright environment was measured to 1450 lux next to the display, dim environment was measured to 500 lux, dark environment was measured to close to zero lux, and a gray display image used 50% white level.

On the device, eight predefined crosshairs was shown, see Figure 6.1a, and the user was asked to touch each of the crosshairs, once with their right index finger, as if it was a button displayed on a regular touchscreen. After the user had touched all these crosshairs once, this step was repeated once again using the same state.

When the user had touched 16 times on the crosshairs in the same state, the test moved on to the next one. The first section of the user test was finished when the user had finished all the nine states.

For the second section of the user test, the user was shown a simple application with a number of buttons on the display, see Figure 6.1b. The user was asked to familiarize herself with the device by pressing the buttons. After a button was pressed, it was visually indicated that it had been activated and after a slight delay, it disappeared. In order to activate a button, the button had to be pressed for a short time. This was decided since the user might otherwise accidentally activate buttons and therefore be confused. When all buttons have been cleared from the screen, the buttons appear once again. This *training process* continued until the user said they were satisfied.

When the user had finished the training process, the test continued by repeating the same steps that was made in the first section of the user test. By doing this training process, it was hoped that the intuitivity and how close it meets the user's expectations of a touchscreen device could be evaluated.

In total, every participant would touch the targets 288 times (8 crosshairs $\cdot$ 2 times $\cdot$ 9 test states $\cdot$ 2 executions).

Finally, the user was asked to fill in a questionnaire that asked about the users background regarding touchscreens as well as if she perceived any differences before and after the training process. The survey's questions and all the answers are listed in Appendix B.

6.2 Preparation

The room that was used for the user test was the same room that was used for all experiments described in Sections 5.4, 5.6 and 5.7.1. For the evaluation, the window pane in the door was firmly sealed to minimize external factors. The touchscreen device was placed on the table and markings was placed on the floor to indicate the user's position. See Figure 6.2 for a photo of the setup. Furthermore, a video camera was positioned so that the user's hand was clearly seen from the side during the test execution. A lux-meter was placed just next to the device's panel and a thermometer was also attached to the device.

Finally, since most of the participants were not proficient in English, help was received to translate all the questions as well as the written instructions into Japanese.

6.3 Execution

Except for the written instructions that were translated into Japanese, the subjects were also able to orally ask questions. However, answers could only be provided in basic Japanese (JLPT 3 level).

The empirical evaluation was first carried out with a pilot study of three test subjects. After the pilot study, some minor adjustments were made on the test. First of all, the camera position was changed from the opposite side of the user to the side of the user. It was also deemed superfluous to take a whole posture picture of the subject as it disturbed the workflow of the study, as only a single camera was provided which meant it was needed to be detached from the tripod and once again mounted for the recoding. Furthermore, some questions was changed to

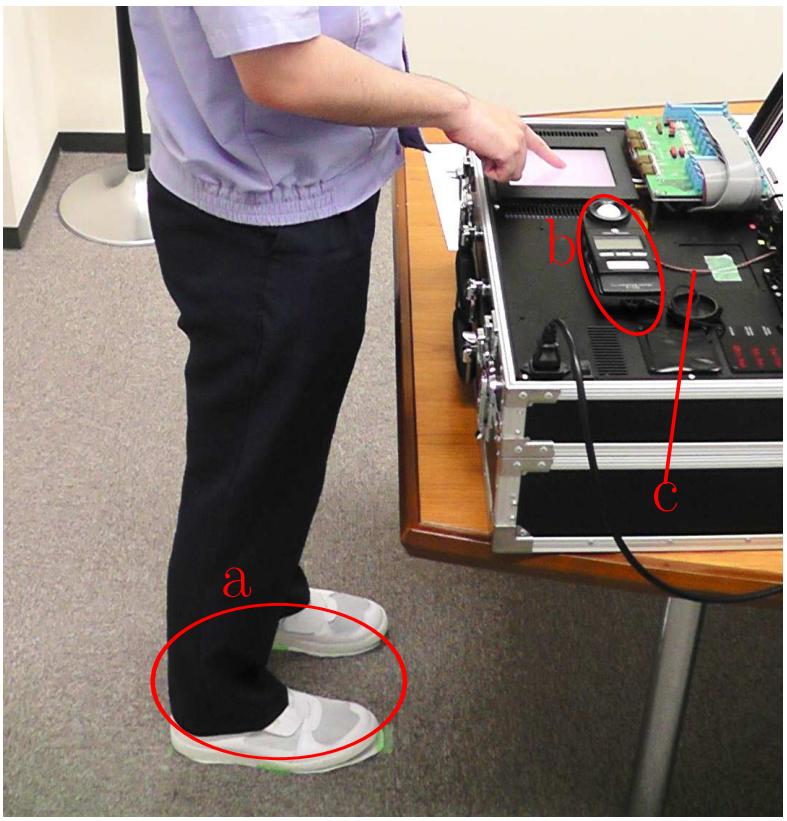


Figure 6.2: The setup for the evaluation and one of the test subject's posture during the test. Notice that the subject is standing on the marked position on the floor (a), the lux-meter lying next to the display (b), as well as the thermometer attached to the device (c).

ask the user to provide more information. For example, question 6 (see Appendix B): "Did the training phase change the way you touched the second time?" was changed to "Did the training phase change the way you touched the second time? *If so, what changed*?". Minor changes were also made to the applications used to decrease any confusions and streamline the test. Such changes include decreasing the time the buttons had to be pressed to activate them and showing a black screen between each test state.

Chapter 7

Results

This chapter will present some performance results followed by the results of both the informal and the formal evaluation of our proposed algorithm.

7.1 Code

The release build of the application was able to run each frame on average in 12 ms as can be seen in Figure 7.1. The settings used are shown in Table 7.1. Except for a few peaks from time to time, the code has no problem in running within 60 fps (under 16 ms per frame). The cyclic pattern in Figure 7.1 are due to the fact that there are no touchpoints in the beginning and end of the test video files and the labeling process is one of the heavier operations in the algorithms.

Parameter	Value
binarization threshold	220
labeling kernel size	8
k	2.8
minimum touch area	100
maximum touch area	5000
minimum contour size	5
maximum contour size	200
filled contour ratio	0.5
maximum width and height ratio	3.0

Table 7.1: The settings used for the test run shown in Figure 7.1.

.

To optimize the code, the profiling tool CodeXL described in Section 5.2.2 was used. As can be seen in Table 7.2, the selective normalization process (described in Section 5.5.3) is among the heaviest operations. Since doing operations using floating point types is much more computationally expensive than using integers stored in bytes, the application used the char types which are single byte integer data types in C++ as its main data type. However, the histogram normalization needs to transform the data at least once into floating points and back again, which is a very expensive operation. This is also true for the *smoothVerticalStrips* function.

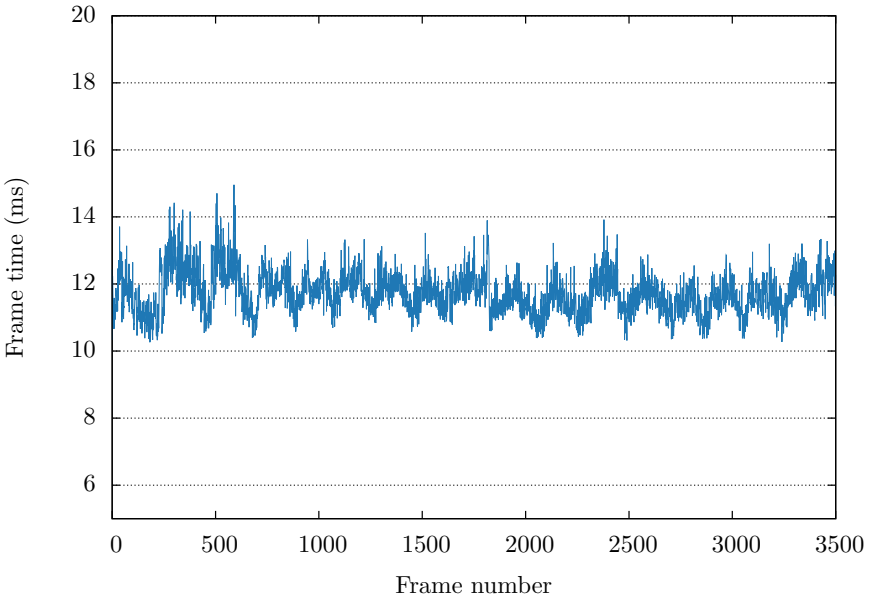


Figure 7.1: The frame time for each frame taken for each test from experiment 2 described in Section 5.6.

Function	Samples
SelectiveNormalization	5400
SelectiveNormalization::shift_histogram	2378
Calibration::smoothVerticalStrips	597
Labeling::scanImageAssignLabels	488
Debug::drawDebugStack	393
Labeling::findContours	227
FingerFeatures::calculateRotation	52
FingerFeatures::calculateCentroid	3
Tracking::getSortedNeighbours	3
Calibration::calibrateCorrection	2
FingerFeatures::calculateAndAssignTouchPointRotation	1
Labeling	1
Labeling::trimBadAreas	1
Tracking::removePreviousPointsGoingUp	1
Tracking::setTouchStates	1

Table 7.2: The results after running the profiler on all the tests in experiment 2 described in Section 5.6.

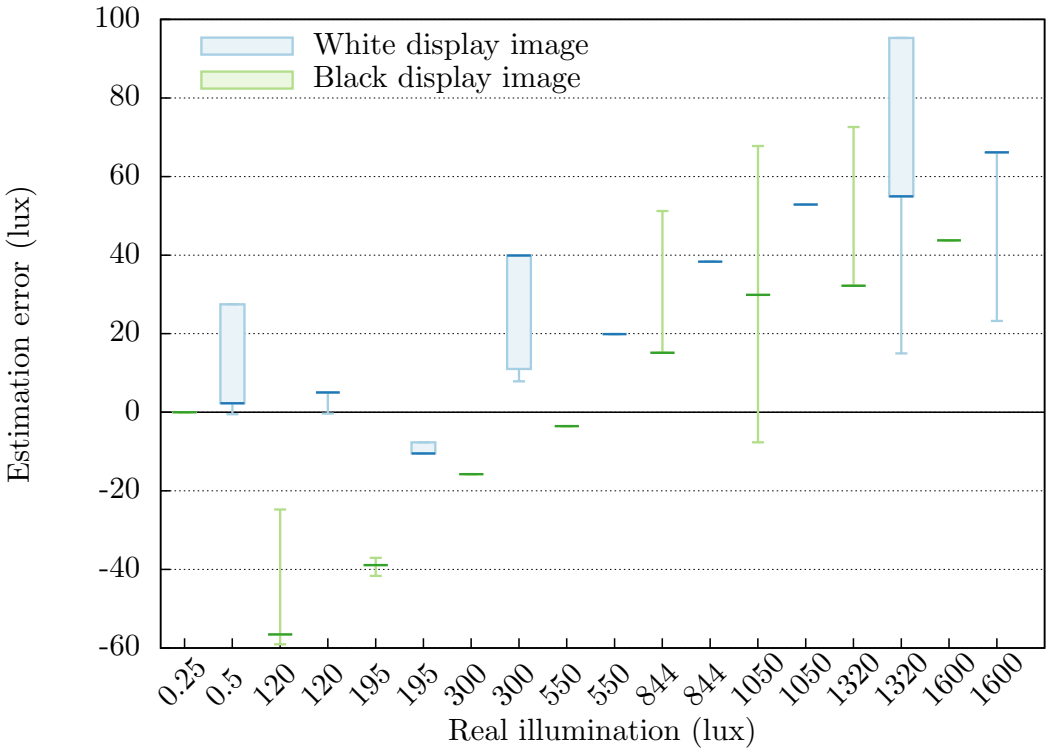


Figure 7.2: The figure shows the estimation error for various illumination values and display image colors represented as a box plot. The whiskers represent the minimum and maximum values respectively.

7.2 Algorithm

In this section the evaluations and results of our algorithm will be presented.

7.2.1 Informal evaluation

In order to evaluate the illumination estimation process explained in Section 5.5.2 several sensor data videos were captured in different lighting environments. The videos were recorded while a person was touching and blocking parts of the screen using two fingers. Despite this interference – given the fact that the estimation algorithm uses a history to prevent dips – the results were good, as can be seen in Figure 7.2.

As described in Section 5.5.3 and illustrated in Figure 5.8 the touched contact areas in varying lighting and display image conditions were clear and usable. Figure 7.3 compares our algorithm with one that does not consider either the ambient illumination nor the displayed image. Our algorithm generated 105 (82%) true touchpoints (i.e., touchpoints registered where the user actually made a touch) which compared to the other is a 80% improvement. It also generated only 18 (14%) false touchpoints (i.e., touchpoints registered that where *not* an actual touch by the user) which is a 70% improvement.

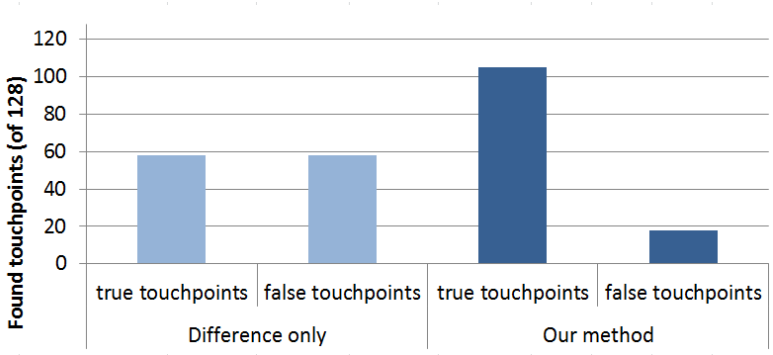


Figure 7.3: Results from our algorithm compared to using only the difference between backlight on and off. This result is based on 128 touches ranging from 0 to 1600 lux on black and white display images respectively. Both tests use the same global threshold segmentation and connected components method for finding the touchpoints.

7.2.2 Empirical evaluation

A lot of data was collected during the user test: recorded camera video of the participants' hands, room temperature, the questionnaires, and the actual sensor data. The users have together touched the crosshairs a total of 4608 times in varying lighting and display conditions.

But, what defines a single touch? A user might not keep the fingertip at a constant position at the screen. At what point in time, from the moment the finger touches the surface of the touchscreen, to the moment it releases it, should the coordinates of the finger be registered as the data? In our case, it was decided that the *middle element* in the touch sequence between *touch down* and *touch up* should count as the actual data. Also, touch sequences of three frames or less were all discarded. This was done to avoid unintentional touches caused of shadows or other interferences.

An evaluation program was written for the purpose of generating and extracting the results from the raw sensor videos. It automatically processed all captured sensor videos of all participants in all light and display states, both before and after the training phase. For every state, the program mapped the user's touchpoint (calculated with our algorithm) to the corresponding crosshair and output the coordinates for them respectively. That output is used for all the graphical representations in this section which are presented below.

The scatter diagrams in Figure 7.4 shows all 16 users' touchpoints during the various test states. As can be seen, all users tend to touch slightly to the lower right of the actual target center. For a closeup of a specific crosshair, Figure 7.5 shows the touchpoints scattered together with a histogram of the distance from the target center.

Depending on the participant, result may vary greatly. In Figure 7.6 a comparison between two users are shown; one is touching with high true touchpoints result and the other with not low true touchpoints. For an accuracy and registered touch ratio comparison with all the participants see Figure 7.7.

The results of the empirical evaluation are shown in Table 7.3. In general, the results lies in range of the previous informal test (Section 7.2.1), however it has slightly fewer true touchpoints but also fewer false touchpoints. Comparing before and after the subjects did the interactive training process, described in Section 6.1, it is shown that there are only a few percentage differences, both increasing and decreasing depending on what is measured, suggesting that the training process had no major impact.

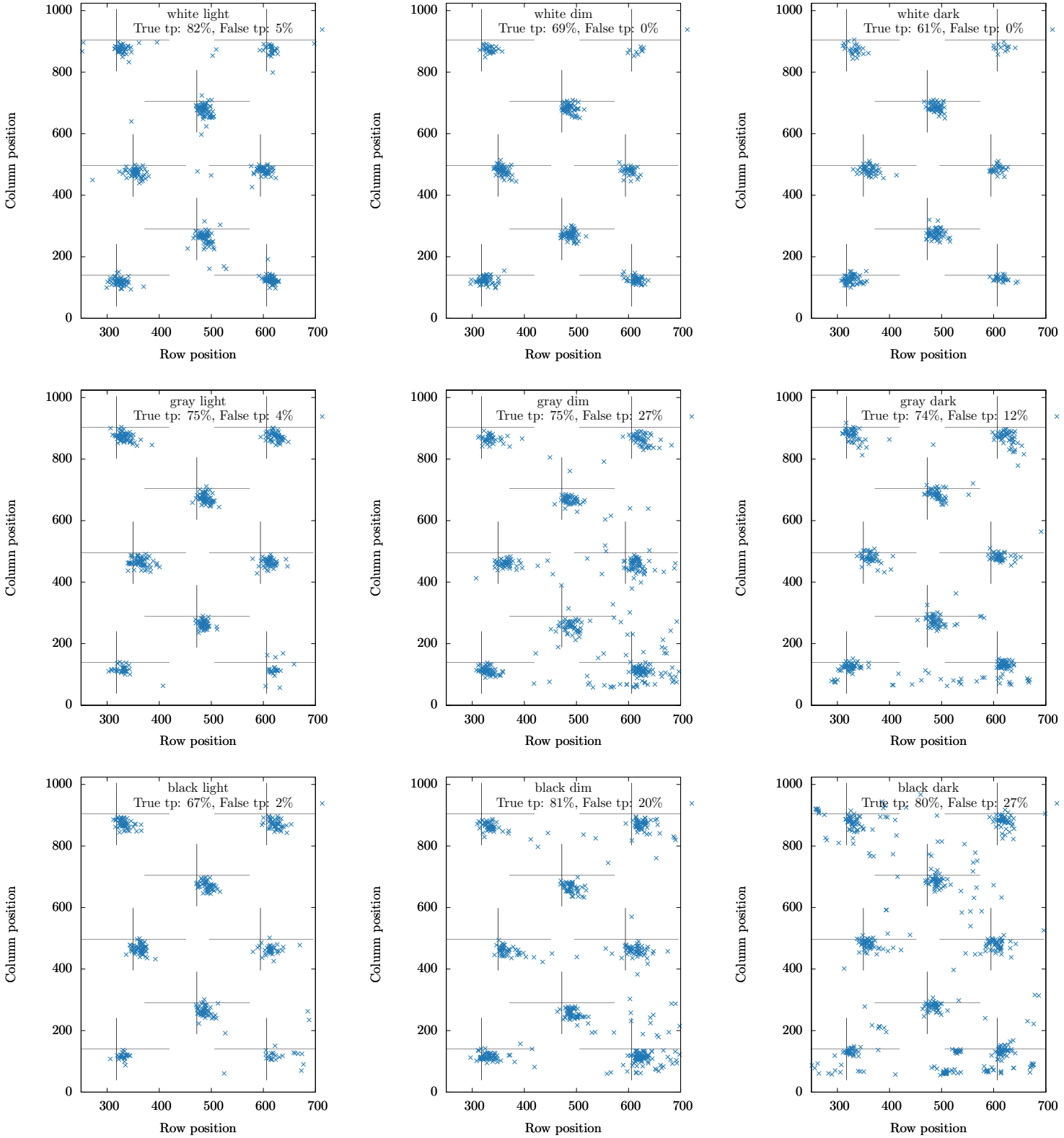


Figure 7.4: Showing a diagram of all registered touchpoints for all 16 participants during the empirical evaluation. The diagrams are separated by states. The title of each diagram states the display and light condition as well as the true- and false touchpoints (tp) ratio. A low true touchpoints ratio suggests that a lot of touches were missed by the algorithm. A high false touchpoint ratio suggests that algorithm registered a lot of touchpoints that were not actually touched by a user.

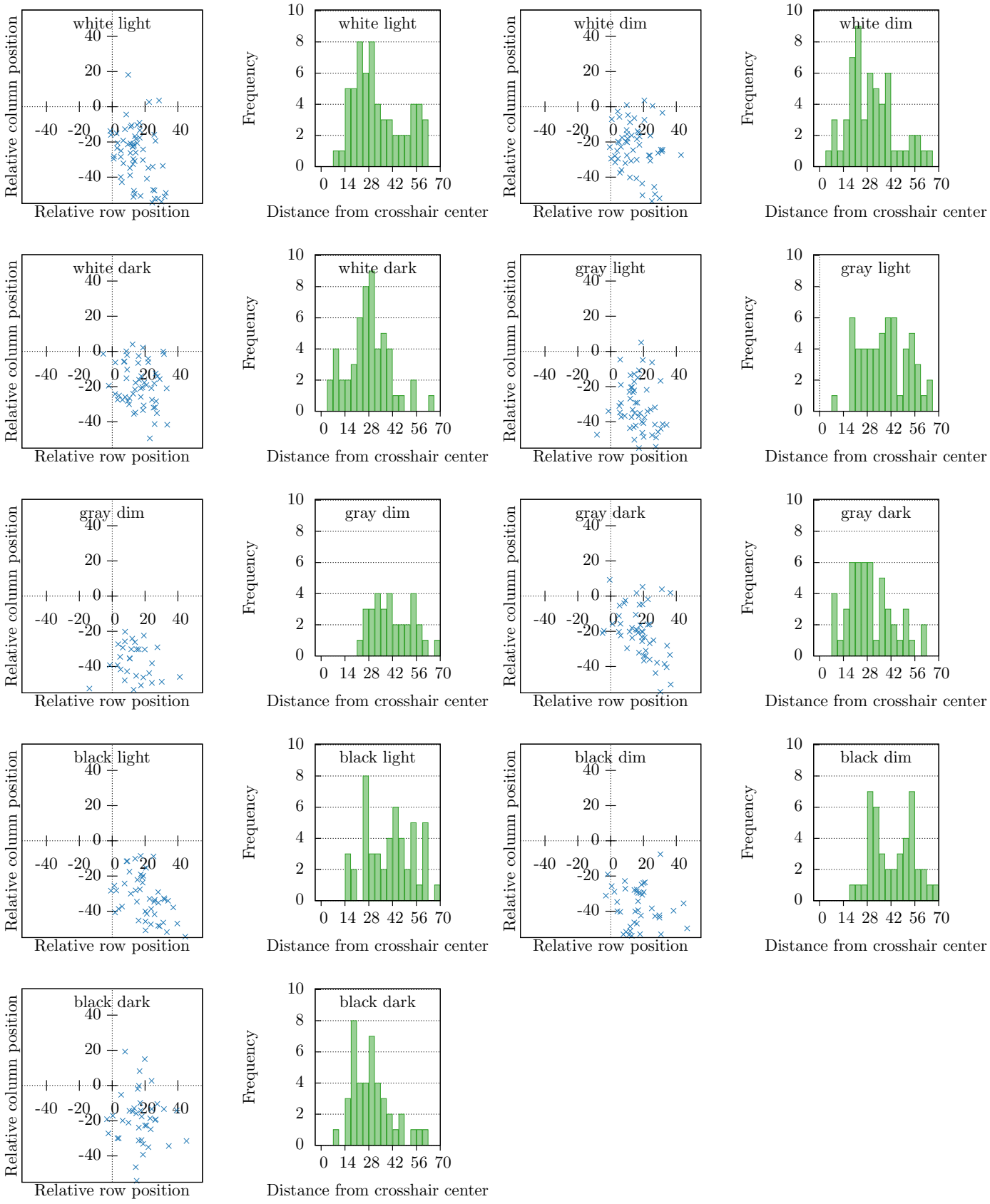


Figure 7.5: Scatter diagram of all 16 users' touchpoints on a selected crosshair target (target #3 in Figure 6.1a). To the right the distance from the center can be seen in histograms.

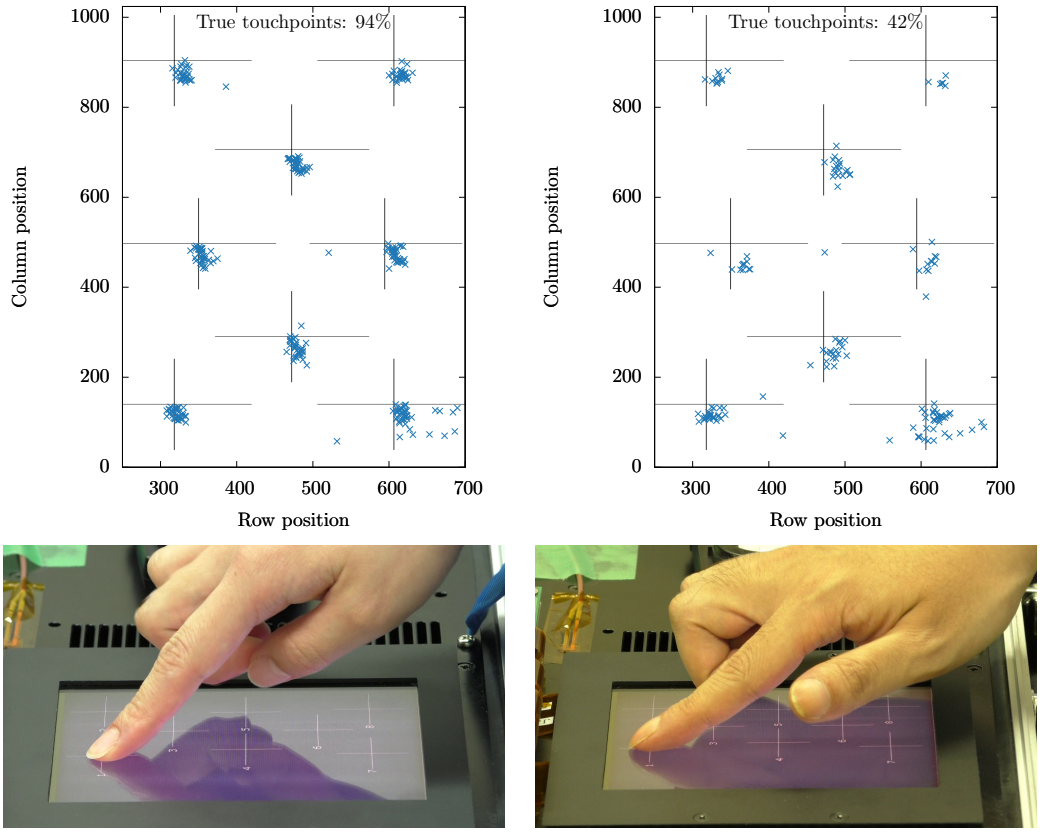


Figure 7.6: True touchpoint comparison between test subject #9 and #3. The upper row shows the touchpoints which are collected from all nine states. The lower shows an example frame from the video footage of the two users. Observe how the "good" user has a higher incident angle as well as a larger touch area.

The evaluation was executed with 16 test subjects, six were female and their ages ranged from 24 to 37 with 31 being the average. Almost all of them had some previous experience with touchscreens, usually around 2-4 years of experience with smartphones, however most of them had none to very little experience with optical touchscreens. However, only three participants reported that they had any experience of *optical* touchscreen devices. Most of them were right handed where two used both hands and one mainly used the thumb to interact with regular smartphones. The average time for a subject to carry out the test was nine and a half minutes and the average temperature of the device during the test was 24.5 decrees Celsius. More than half of the participants reported that their way of interacting with the device changed slightly, generally touching more firmly and carefully, and the rest of them reported that they did not change. The full results of the questionnaire with further comments from the participants are shown in Appendix B.

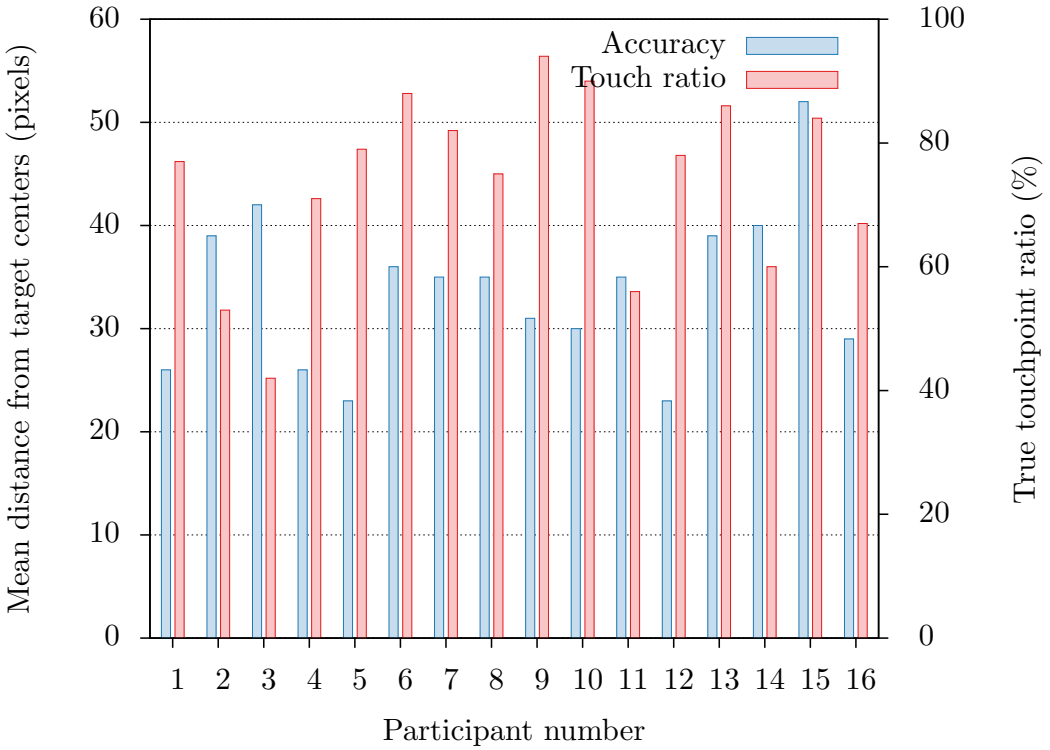


Figure 7.7: All participants' mean distance from all crosshair centers (left box) as well as the touch ratio (right box) visualized as a box diagram.

	Actual touches	Registered touches	True touchpoints	False touchpoints
Before	2304	1970 (85%)	1728 (75%)	242 (11%)
After	2304	1977 (86%)	1694 (74%)	283 (12%)
Total	4608	3947 (86%)	3422 (74%)	525 (11%)

Table 7.3: The table shows the actual number of touches the user was supposed to do, the total number touches the algorithm registered, and finally the number of true touchpoints as well as the number of false touchpoints. It is spearated into before and after the training session as well as the sum of them both. All relative numbers are compared to the associated actual number of touches.

Chapter 8

Discussion and future work

8.1 Empirical evaluation

Although we only had basic proficiency in Japanese and the test subjects had very little or no proficiency in English, the test mostly went smoothly. Most of the test subjects reported that the instructions were easy to follow and we believe that there were no data loss or disturbance caused by a language barrier.

Since the algorithm was mostly developed with test data taken by us touching the device, or at a few occasions, our supervisors, who had previous experiences with developing touchscreen devices, the parameters had been optimized for an experienced hand. This is especially observable when comparing the informal test with the evaluation. Because the informal test was done by us, the parameters had been set loose. However, they were more strictly set during the evaluation. As observed in Chapter 7, this resulted in fewer true touchpoints, but also fewer false touchpoints in the evaluation.

A minor cause of noise for the results could be how the test subject was standing over the device. Depending on how much the user leaned over the device, the illumination could decrease as much as 20%, and thus, the measured illumination for the lighting might not actually be exactly the same between each test subject. However, the illumination mostly matters in very low ranges, 100-300 lux, and it is therefore believed that this is a minor cause of error.

How we define a touch during the evaluation is described in Section 7.2.2, i.e., the coordinate in the middle of the sequence of touch coordinates from when a user touches down until release. However, this could have implications on where the touch locations are calculated. On one hand, we make sure that the touch is firmly pressed down, i.e., we prevent false touchpoints that could otherwise be registered when the user moves to and from the crosshair. For example, this would be a problem if we only used the location from the touch when the user lifts its finger. But on the other hand, if the majority of the touches in one sequence are located far away from the crosshair, the final location will be positioned further away from the center. Another way of defining touch could be to take the touch in the sequence that has the *greatest area*, i.e., greatest pressure applied to it, assuming this better models the users intent of indicating a touch. This also better mimics the mechanics of a physical button, which generally require some force applied to the touch.

As shown in Section 7.2.2, there were no significant difference between the tests before and after the training session which implies that the training session did not change their behaviour. We included the training process to test whether our system was simple and intuitive to use. Since most of the subjects had significant experience with capacitive touch technologies (used in

the majority of today's mobile phones) we supposed that all users would initially interact with the system in a similar fashion. If the users however would change their behaviour and have significantly different results after the training session, we could assume that our device require another way of interacting. Since the users' results did not change, one conclusion could be that the system is indeed similar to capacitive technologies. However, it could also be simply that the training process did not actually train the subjects enough. Since there is no feedback when the subject touches the crosshairs, it is feasible that they quickly forgot what they learned in the process. However, reading the results of the questionnaires, shown in Appendix B, some test subjects reported that they did in fact change their way of touching, which was also observed on the recorded videos. This could mean that their changed behaviours actually did not have any impact.

As can be seen in both Figures 7.4 and 7.5, the calculated touchpositions tended to be positioned slightly down to the right of the crosshairs. This confirms that a better and more accurate model of the users intent should be used as mentioned previously in Section 8.3. The reason why they are located down to the right of the crosshairs could be because the users target with their upper part of their finger, which results in a touch area reaching down and right (as the subjects used their right hand) from which the geometric center is calculated. Furthermore, the fact that the subjects used their right hand during the tests explains why there is a higher amount of faulty and incorrect touch positions in the southwest quadrant of the touchscreen, since most of the time, the hand simply shadows this area more than the rest of the screen. This is especially observed in the gray and black test states, which are heavily dependent on the shadow information.

8.2 Hardware

A lot of problems, especially ambient light sensitivity and the black screen problem (explained in Section 3.2), arise because of the sensors capture the full visible spectrum of light, not limited to a certain spectrum.

However, during the internship period we did not have the possibility to change nor modify the hardware in any way. Also, we did not have access to modify the embedded software in the touchscreen which, among other things, controlled the voltage to the backlight. Nevertheless, for the purpose of future work, we will discuss some potential hardware modification suggestions.

One way to solve the black display problem would be to change the way how the touchscreen controller captures the images. Currently, as described in Chapter 3 the device captures two images every frame; one taken with the backlight turned on, and one taken with the backlight turned off. By capturing a *third* image every frame – one that is taken with the backlight turned on while forcing the display to show a white image – it would insure that touchpoint acquisition always would be possible. Another way would be to simply capture the backlight-on-state while forcing a white output in order to keep the number of images captured per frame to two. Since we were unable to change the capture controller for the touchscreen, we tried to implement the *flash-a-white-image-during-capture*-method in software. In the implementation we had to balance the capture frequency vs. the actual displayed images on the screen. In order to keep a relatively high capture frequency we have to force the screen white often, but then the user will notice the flashing. Even though flashing the screen white every 10th frame it was still perceived annoying and we concluded that a software implementation of this was not feasible.

Another way to avoid the black display problem would simple be to increase the backlight emission, so that more light could be transmitted through the LCD layer even while it is in off-state. However, as discussed before, this will increase the power consumption greatly.

As discussed in the analysis of Section 5.4, a dead-zone is present. It arises when the ambient light falling upon the screen has the same illumination as the backlight's reflection back upon itself (either on the glass layer or from a finger). One way to solve this could be to check for this scenario – using for example the illumination estimation procedure described in Section 5.5.2 – and in the hardware controller change the voltage to the backlight so that the dead-zone disappears.

In order to achieve robustness for changes in ambient light and avoid the black screen problem, the currently best way seems to be to use some kind of infrared backlight. As mentioned in Previous work (Section 2.2.1) Tanaka et al. [2011] describes an optical touchscreen with infrared backlight that works well.

Some other similar in-cell optical touchscreens have built-in noise reduction circuits for increasing the signal-to-noise ratio. The prototype developed by Abileah and Green [2007] also features built-in automatic gain control and dynamic range adjustments. Our prototype did not have any of these features which resulted in a very low dynamic range in dark environments (as can be seen in Figure 5.4).

During this study we have not taken different wavelengths of light into consideration. We have not examined for example how the sensors react to red light compared with blue light. This has been left out as future work.

8.3 Algorithm

Since each sensor's value range is normalized in the calibration process, a global image segmentation operator, such as global thresholding, should be sufficient for the algorithm. It might have been interesting to try to implement and use a local segmentation algorithm such as the watershed algorithm or the clustering-based thresholding "Otsu's method" [Otsu, 1979]. However, as discussed in Section 8.2, the signal from the sensors have a rather low dynamic range as well as a high level of noise, therefore, the accuracy might not increase significantly. Furthermore, a global threshold is as efficient as it can be, using only a comparison and an assignment per pixel. However, local variants' complexities usually increase the computational time at least by the number of inspected neighbouring pixels.

We also experimented with the built-in OpenCV function to calculate contours. As described in Section 4.2, this function was reported successfully used to detect touch points. However, we found out that it is both difficult to use (it required tedious fine-tuning of many parameters) and also had significantly increased computational times compared to our hand-written functions.

Finally, to calculate the actual touch position, the algorithm only calculates the geometric center of the touched area. However, as Holz and Baudisch [2011] shows in their report, this method poorly reflects the users intended touch position. Therefore, it would be interesting to see if any of the better performing methods that was proposed could be adapted to in-cell devices that better models the users' intent.

Chapter 9

Conclusion

In this project we proposed an algorithm for acquiring touchpoints in varying ambient illumination and display images. The prototype captures two images per frame; one taken with the backlight turned on, and one taken with the backlight turned off. These are sent to the host system, in our case a PC, and calibrated with pre-captured reference images. Furthermore, the algorithm estimates the ambient illumination and uses it together with the displayed image in order to get normalized sensor values independent of external factors. This image is segmented using a global threshold and the touched areas are found and labeled by a custom connected-component method. The areas are then used for calculating touchpoint coordinates. Methods for calculating both finger rotation and pressure was successfully implemented and the findings also suggest that the incident finger angle could be obtained. However, the experiments showed that accurately obtaining height of the finger is, with this technology, unlikely since the reflected light is scattered too quickly. Although the exact 3D position is not obtainable, it would be possible to implement coarse swiping guestures.

An informal test suggested that using our proposed approach, the range of usable conditions was extended considerably compared to a conventional approach. Furthermore, a formal empirical evaluation was conducted as a usability test which further confirms the previous findings. However, this technology is still not ready for consumer market as the accuracy and robustness still does not reach the quality that is expected of modern touch-enabled devices.

To summarize, further improvements needs to be done on the hardware for this technology to be usable in mass-produced products. Further noise reduction techniques need to be adapted for the hardware and the problem with black display outputs needs to be solved by hardware rather than software, e.g., using only the infrared spectrum for sensing light.

Bibliography

Adi Abileah and Patrick Green. Optical sensors embedded within amlcd panel: Design and applications. In *Proceedings of the 2007 Workshop on Emerging Displays Technologies: Images and Beyond: The Future of Displays and Interacton*, EDT '07, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-669-1. doi: 10.1145/1278240.1278247. URL `http://doi.acm.org.proxy.lib.chalmers.se/10.1145/1278240.1278247`.

Adi Abileah, Willem Boer, Terrance Larsson, Tom Baker, Scott Robinson, Roy Siegel, Noah Fickenscher, Brent Leback, Terry Griffin, and Pat Green. 59.3: Integrated optical touch panel in a 14.1 amlcd. In *SID Symposium Digest of Technical Papers*, volume 35, pages 1544–1547. Wiley Online Library, 2004.

A. Agarwal, S. Izadi, M. Chandraker, and A. Blake. High precision multi-touch sensing on surfaces using overhead cameras. In *Horizontal Interactive Human-Computer Systems, 2007. TABLETOP '07. Second Annual IEEE International Workshop on*, pages 197–200, Oct 2007. doi: 10.1109/TABLETOP.2007.29.

Willem Boer, Adi Abileah, Pat Green, Terrance Larsson, Scott Robinson, and Tin Nguyen. 56.3: Active matrix lcd with integrated optical touch screen. In *SID Symposium Digest of Technical Papers*, volume 34, pages 1494–1497. Wiley Online Library, 2003.

C. Brown, B. Hadwen, and H. Kato. A 2.6 inch vga lcd with optical input function using a 1-transistor active-pixel sensor. In *Solid-State Circuits Conference, 2007. ISSCC 2007. Digest of Technical Papers. IEEE International*, pages 132–592, Feb 2007. doi: 10.1109/ISSCC.2007.373623.

Chris Brown, David Montgomery, Jean-Luc Castagner, Hiromi Kato, and Yuuichi Kanbayashi. 31.3: A system lcd with integrated 3-dimensional input device. *SID Symposium Digest of Technical Papers*, 41(1):453–456, 2010. ISSN 2168-0159. doi: 10.1889/1.3500490. URL `http://dx.doi.org/10.1889/1.3500490`.

Bill Buxton et al. Multi-touch systems that i have known and loved. *Microsoft Research*, 56: 1–11, 2007.

W. Buxton, E. Fiume, R. Hill, A. Lee, and C. Woo. Continuous hand-gesture driven input. In *Proceedings of Graphics Interface '83, 9th Conference of the Canadian Man-Computer Communications Society*, pages 191–195, Edmonton, May 1983. URL `http://www.billbuxton.com/gesture83.html`.

Jennifer Colegrove. The state of the touch-screen market in 2010. *Information Display*, 3(10): 10, 2010.

Software Freedom Conservancy. About git. http://opencv.org/about.html, 2014. Accessed: 3 Apr 2014.

Luigi di Stefano and Andrea Bulgarelli. A simple and efficient connected components labeling algorithm. In *Proceedings of the 10th International Conference on Image Analysis and Processing*, ICIAP '99, pages 322–, Washington, DC, USA, 1999. IEEE Computer Society. ISBN 0-7695-0040-4. URL http://dl.acm.org/citation.cfm?id=839281.840794.

Paul H. Dietz and Benjamin D. Eidelson. Surfaceware: Dynamic tagging for microsoft surface. In *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction*, TEI '09, pages 249–254, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-493-5. doi: 10.1145/1517664.1517717. URL http://doi.acm.org.proxy.lib.chalmers.se/10.1145/1517664.1517717.

Vitaly Friedman. Microsoft surface technology. http://w5.cs.uni-sb.de/teaching/ws0708/AIHCI/, 2007. Accessed: 3 Apr 2014.

Rafael Gonzalez and Richard Woods. *Digital image processing*. Prentice Hall, Upper Saddle River, N.J, 2008. ISBN 9780131687288.

Guinness World Records Corporate. Most patents held by a person. http://www.guinnessworldrecords.com/world-records/11000/most-patents-held-by-a-person, 2011. Accessed: 31 Mar 2014.

Jefferson Y. Han. Low-cost multi-touch sensing through frustrated total internal reflection. In *Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology*, UIST '05, pages 115–118, New York, NY, USA, 2005. ACM. ISBN 1-59593-271-2. doi: 10.1145/1095034.1095054. URL http://doi.acm.org/10.1145/1095034.1095054.

Steve Hodges, Shahram Izadi, Alex Butler, Alban Rrustemi, and Bill Buxton. Thinsight: Versatile multi-touch sensing for thin form-factor displays. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, UIST '07, pages 259–268, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-679-0. doi: 10.1145/1294211.1294258. URL http://doi.acm.org/10.1145/1294211.1294258.

Christian Holz and Patrick Baudisch. Understanding touch. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, CHI '11, pages 2501–2510, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0228-9. doi: http://doi.acm.org/10.1145/1978942.1979308. URL http://doi.acm.org/10.1145/1978942.1979308.

S. Hotelling, J. Strickon, and B. Huppi. Multipoint touchscreen, May 11 2006. URL https://www.google.co.jp/patents/US20060097991. US Patent App. 10/840,862.

Itseez. About opencv. http://opencv.org/about.html, 2014. Accessed: 2 Apr 2014.

Myron W. Krueger, Thomas Gionfriddo, and Katrin Hinrichsen. Videoplace—an artificial reality. *SIGCHI Bull.*, 16(4):35–40, April 1985. ISSN 0736-6906. doi: 10.1145/1165385.317463. URL http://doi.acm.org.proxy.lib.chalmers.se/10.1145/1165385.317463.

SK Lee, William Buxton, and K. C. Smith. A multi-touch three dimensional touch-sensitive tablet. *SIGCHI Bull.*, 16(4):21–25, April 1985. ISSN 0736-6906. doi: 10.1145/1165385.317461. URL http://doi.acm.org/10.1145/1165385.317461.

MME-250: metal halide light source. Schott moritex corporation. https://www.schott-moritex.co.jp/products/mvs/property.php?c_code=A-0489, 2014. Accessed: 17 Jul 2014.

Microsoft Corp. Pixelsense. http://w5.cs.uni-sb.de/teaching/ws0708/AIHCI/, 2012. Accessed: 3 Apr 2014.

Microsoft Corp. Touch injection. http://msdn.microsoft.com/en-us/library/windows/desktop/hh802898(v=vs.85).aspx, 2014. Accessed: 17 Jul 2014.

Nobuyuki Otsu. A threshold selection method from gray-level histograms. *Systems, Man and Cybernetics, IEEE Transactions on*, 9(1):62–66, Jan 1979. ISSN 0018-9472. doi: 10.1109/TSMC.1979.4310076.

J. Preece, Y. Rogers, and H. Sharp. *Interaction design: beyond human-computer interaction*. J. Wiley & Sons, 2002. ISBN 9780471492788. URL http://books.google.co.jp/books?id=AFFGAQAAIAAJ.

Azriel Rosenfeld and John L. Pfaltz. Sequential operations in digital picture processing. *J. ACM*, 13(4):471–494, October 1966. ISSN 0004-5411. doi: 10.1145/321356.321357. URL http://doi.acm.org/10.1145/321356.321357.

Semiconductor Energy Laboratory Co., Ltd. Creating technology. Brochure, Hase, Atsugi-shi, Kanagawa, Japan, 12 2013.

L.G. Shapiro and G.C. Stockman. *Computer Vision*. Prentice Hall, 2001. ISBN 9780130307965. URL http://books.google.se/books?id=FftDAQAAIAAJ.

Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image Processing, Analysis, and Machine Vision*. Thomson-Engineering, 2007. ISBN 049508252X.

Hikaru Tamura, Toshiki Hamada, Takashi Nakagawa, Takeshi Aoki, Masataka Ikeda, Munehiro Kozuma, Yoshiyuki Kurokawa, Takayuki Ikeda, Koji Moriya, Yoshiharu Hirakata, Nozomi Kamata, Tsutomu Murakawa, Jun Koyama, Shunpei Yamazaki, Katsuaki Tochibayashi, Kenichi Okazaki, and Masayuki Sakakura. 50.2: High reliable in-ga-zn-oxide fet based electronic global shutter sensors for in-cell optical touch screens and image sensors. *SID Symposium Digest of Technical Papers*, 42(1):729–732, 2011. ISSN 2168-0159. doi: 10.1889/1.3621430. URL http://dx.doi.org/10.1889/1.3621430.

Kohei Tanaka, Hiromi Kato, Yasuhiro Sugita, Naru Usukura, Kazuhiro Maeda, and Chris Brown. The technologies of in-cell optical touch panel with novel input functions. *Journal of the Society for Information Display*, 19(1):70–78, 2011. ISSN 1938-3657. doi: 10.1889/JSID19.1.70. URL http://dx.doi.org/10.1889/JSID19.1.70.

David J Tulbert. 31.4: Low cost, display-based, photonic touch interface with advanced functionality. In *SID Symposium Digest of Technical Papers*, volume 36, pages 1222–1225. Wiley Online Library, 2005.

L. Vincent and P. Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 13(6):583–598, Jun 1991. ISSN 0162-8828. doi: 10.1109/34.87344.

Geoff Walker. Display week 2012 review: Touch technology. *Information Display*, 9:12, 2012.

Geoff Walker. Touch technologies tutorial. `http://www.walkermobile.com/PublishedMaterial.htm`, 2013. Accessed: 31 Mar 2014.

Geoff Walker and Mark Fihn. Lcd in-cell touch. *Information Display*, 27:8–14, 2010.

Feng Wang, Xiangshi Ren, and Zhen Liu. A robust blob recognition and tracking method in vision-based multi-touch technique. In *Parallel and Distributed Processing with Applications, 2008. ISPA '08. International Symposium on*, pages 971–974, Dec 2008. doi: 10.1109/ISPA.2008.129.

White Wyman. Method for optical comparison of skin friction-ridge patterns, August 17 1965. US Patent 3,200,701.

Appendix A

Time plan

During our internship at SEL, they wanted us to at some time participate with a paper at a conference. Since the paper submission was shortly after our arrival (about one month later) it was decided upon that we would start focus on reading existing code. However, this conference was later then aborted as the deadline was too soon, and also because the theme of the conference was not matching our research field well, after which we finished porting the existing code into a new code base while it was still fresh in memory. After this, the planning report was written and the actual research for this thesis started.

As the first conference was cancelled, two other conferences with better theme and later paper-submission deadlines were found. It was of SEL's interest that we participated in some of these so the time planning was scheduled around these new conferences. Later, it was decided to join only one conference: MIRU 2014. As mentioned in Chapter 5, the work was made in an iterative fashion, and thus the research and study block runs parallel with the implementation block as we test to implement new methods as they are discovered. Also, it was decided that the main focus of the research should be to investigate the light sensitivity issue as mentioned in Section 1.3 and therefore, this is the main topic of the research for the conference. After the submission deadlines passed for the conference, the other research topics were allowed to be investigated.

Finally, the last month was dedicated for the empirical evaluation and the writing of the thesis.

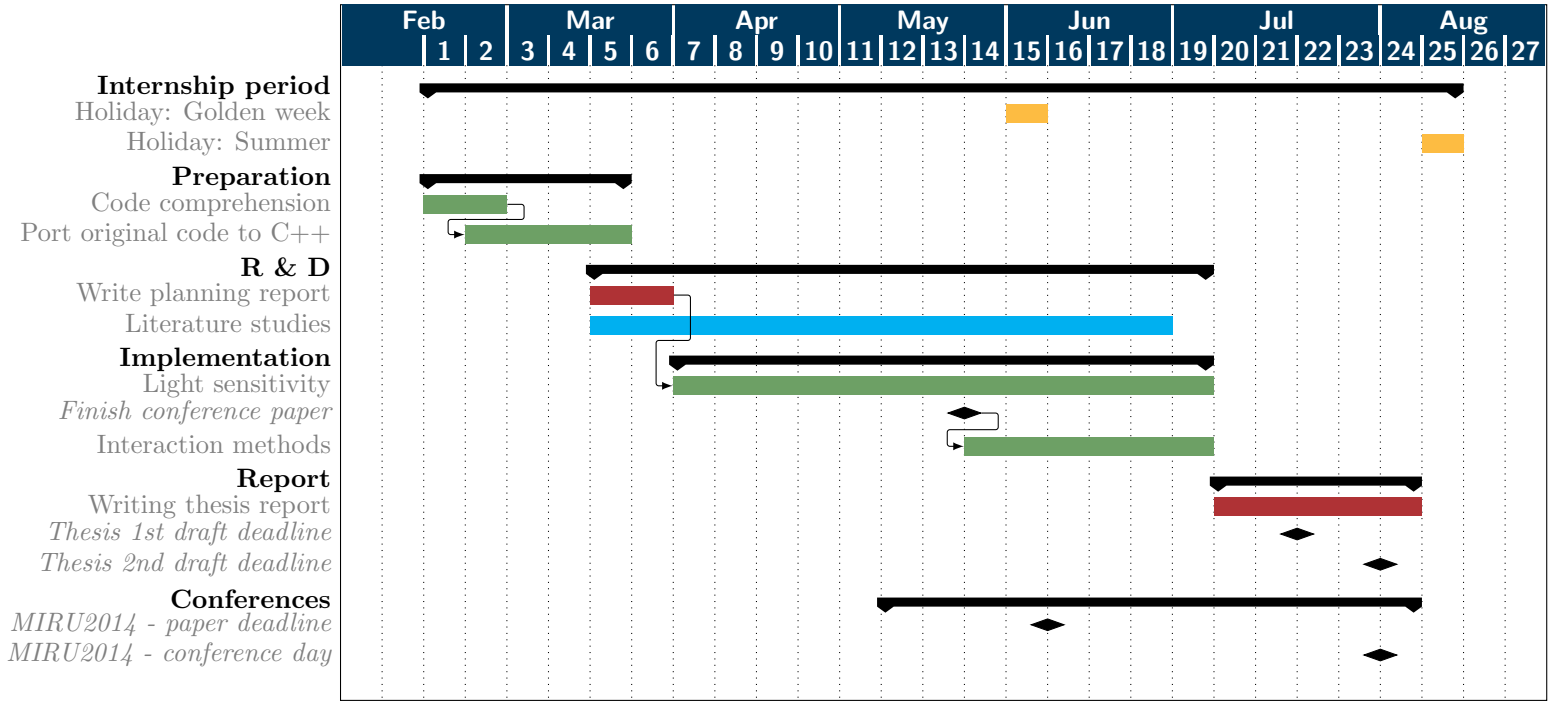


Figure A.1: Time plan visualized as a Gantt chart.

Appendix B

Questionnaire

The questionnaire used for the user test had the following question:

1. Gender
2. Age
3. Which hand do you usually use for touchscreens? (left/right)
4. Previous touchscreen experiences? (e.g. I have used daily since 4 years ago)
5. Previous experience with optical touchscreens?
6. Did the training phase change the way you touched the second time? If so, what changed?
7. What did you think about the test?
8. Were the instructions clear for the test?
9. Other comments?

The results of the questionnaires are summarized in the following tables:

Test subject:	1	2	3	4
Gender:	M	M	M	M
Time:	10	18	11	8
Temp:	27	24.5	24	24
Main hand:	Both	Right	Right	Right
Touchscreen experience:	3 year smartphone. 1 year creating TS software.	3 years smartphone	1 year smartphone	5 years iPhone
Optical touchscreen experience:	Made software 2-3 years ago	None	None	None
Changes after training:	Not much (because exp)	None	None	None
About the test:	Depending on the people, some might want to know why the test was done.	Wanted more training test	Proposes random buttons	Since the reason for the test was not stated, I was puzzled.
Easy instructions:	Easy	Easy	Mostly understood	Easy
Other:				Since the crosshairs don't change when they are pressed, you press harder compared to button test.
Test subject:	5	6	7	8
Gender:	M	M	M	M
Time:	9	9	9	10
Temp:	24	24.5	24.5	24
Main hand:	Right	Right	Right	Right
Touchscreen experience:	4 years ago every day	Used ATM's TS	Daily since 2 years ago	Daily smartphone use since 2 years ago
Optical touchscreen experience:	None	None	None	None
Changes after training:	Pressed more carefully	Pressed more carefully	Don't know	Touch with the "ball of the finger"
About the test:	The prupose of the training was not clear	Since there was no response, touched the crosshairs twice		
Easy instructions:	Easy	Could mostly understand by instruction and looking at screen	Easy	Very easy
Other:		For what purpose was this test made?		

Test subject:	9	10	11	12
Gender:	F	F	M	M
Time:	11	8	7	10
Temp:	24.5	24.7	25	25.1
Main hand:	Mostly R	Right	Right	Right
Touchscreen experience:	Smartphone since 2 years ago	Use smartphone and game console	Everyday since 3 years ago (iPad)	4 years of smartphone. Use both left and right. Usually use thumb.
Optical touchscreen experience:	None	None	None	Yes
Changes after training:	Pressed more slowly and with less force	The time and strength was not known in the beginning. After training, pressed more lightly	Pressed more lightly	Touched more slowly, also was aware of changing angle slightly
About the test:	Training phase (or first test): since the color didn't change, I was worried		"I understood that I did not recognize the difference of my acts, which are touching the panel strongly and touching the panel for a moment longer."	No explanation for why the test was done.
Easy instructions:	Easy	Easy	Easy	Easy
Other:			Randomized buttons would be more fun. Also, humans train easily for fixed positioned buttons.	Would have been easier if the reason for the test was known. Also that the crosshairs don't disappear
Test subject:	13	14	15	16
Gender:	F	F	F	F
Time:	8	7	9	10
Temp:	24.1	24.1	24.8	24.4
Main hand:	Right	Right	Right	Right
Touchscreen experience:	Daily smartphone use	Daily use since 3 years ago	Smartphone since 1 year ago	Daily since 2 years ago
Optical touchscreen experience:	Yes	ATM or train ticket machines? (None)	None	None
Changes after training:	Since no visual feedback, first pressed with force. Pressed more slightly second time.	Pressed longer and more firmly	Mostly none	The force was changed
About the test:			Difficult	The reason for the test was not explained
Easy instructions:	Easy	Firstly little puzzled, but easy	Little difficult but thorough so all right.	Easy
Other:		When the room is dark, the screen is dazzling.		

Appendix C

Conference: The 17th Meeting on Image Recognition and Understanding

As a part of our internship we also wrote a research paper for *The 17th Meeting on Image Recognition and Understanding (MIRU2014)* and went to Okayama prefecture, Japan to present it during a spotlight and poster session. Anything that will be published thorough SEL needs a corresponding patent application, and it was also submitted. The patent is during the writing of this thesis still pending and its patent number and title are listed below.

Filed patent (Japan Patent office)
JP 2014-111367 Program and information processing device

The conference paper and poster are attached as appendices in the following pages.

Image processing for in-cell optical touchscreens

Philip IRRI[1,2,a)] Julian LINDBLAD[1,2,b)] Hikaru TAMURA[1] Jiro IMADA[1] Shunpei YAMAZAKI[1]

1. Introduction

The demand for touchscreen devices has significantly increased every year [4], and touchscreens continue to play an essential role in the market. Consequently, research has focused on innovative technologies such as in-cell optical touchscreens. The use of photodiodes embedded in the pixel-layer gives rise to new possibilities such as hand recognition [1], non-touch gestures [6], bar- and QR-code reading, and fingerprint scanning [2]. However, in-cell optical touchscreens are generally vulnerable to ambient light conditions and do not perform well when a black image is displayed [8]. This paper presents an image processing method for tackling these problems using hardware developed at Semiconductor Energy Laboratory Co., Ltd., Japan (SEL).

2. Related work

Currently, only a few devices utilize in-cell optical technologies, whereas most of them use a certain type of infrared (IR) backlight. The first commercially available device was a trackpad produced by Sharp Corporation in 2009 [8]. To address the problems mentioned in Section 1, Sharp increased the emission rate and sensitivity of the IR light, which negatively affected power consumption. Samsung Electronics developed the SUR40, a large format display device, for Microsoft's PixelSense (previously known as Surface) [5]. However, the operational conditions for this device are restrictive and dependent on ambient lighting [7]. In addition, Toshiba Corporation developed a monitor that detects shadows from ambient light in the visible spectrum rather than in IR light emitted from the backlight [3].

3. Hardware

This paper is based on the In-Ga-Zn-Oxide FET in-cell optical touchscreen developed by Tamura et al. at SEL [6]. This device is a six-inch LCD with 768 (H) × 1024 (V) pixel resolution updated at 60 frames per second (fps) with an embedded light sensor at every fourth pixel. Using a global shutter, the device can capture two sensor images per frame; one image is taken with the backlight turned on, whereas the other with the backlight turned off. These two images are sent to a host system and processed for touchpoint detection.

[1] Semiconductor Energy Laboratory Co., Ltd., 398 Hase, Atsugi-shi, Kanagawa, 243-0036, Japan
[2] Chalmers University of Technology, Gothenburg, Sweden
[a)] pii013@sel.co.jp irri@student.chalmers.se
[b)] jli015@sel.co.jp julianl@student.chalmers.se

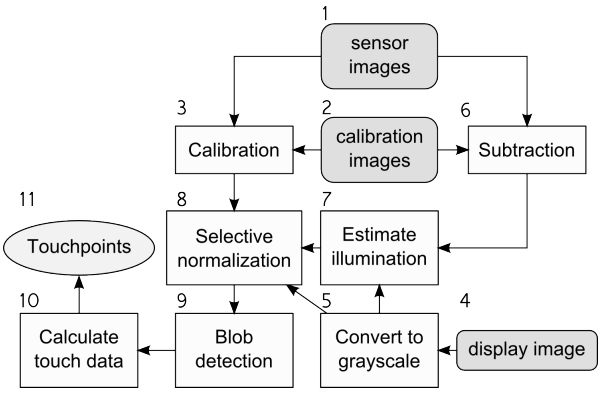


Fig. 1 Overview of proposed algorithm.

4. Algorithm overview

We have developed an algorithm to address the problems outlined in Section 1. The proposed algorithm (Fig. 1) considers both ambient illumination and pixel values of the on-screen displayed image when detecting and calculating touchpoints.

Using two previously captured calibration images taken in the darkest and brightest allowable conditions while the screen displays a white image, the received sensor images are calibrated pixel wise (block 3, Fig. 1). These pixel values are subtracted with the dark calibration image and then scaled such that they lie within the range of the dark and bright calibration images.

A third previously captured calibration image is taken in the darkest allowable condition with a black display image. The sensor image captured with backlight turned off is subtracted with this calibration image to correct sensor offsets (block 6, Fig. 1). This subtracted image is used by illumination estimation (block 7, Fig. 1). Details of the illumination estimation process are provided in Section 4.1. The obtained display image is converted to grayscale (block 5, Fig. 1) and sent together with the estimated illumination to block 8. Here, the calibrated images are normalized using different approaches, depending on input, such that zero and one represent full- and no-touch, respectively. Details of the selective normalization process are provided in Section 4.2.

Finally, touch contours are found using general blob detection methods, and the touchpoint data is calculated. However, this is outside the scope of this paper.

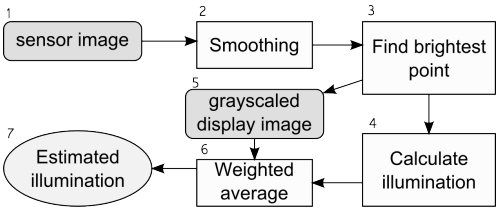


Fig. 2 Overview of the illumination estimation process.

4.1 Illumination estimation

To filter out erroneous sensor values, the sensor image is smoothed before searching for the maximum pixel value. This maximum value is then fed into two functions separately—one each for the black and white display images—that have been approximated on premeasured test data. These functions convert the sensor value into two illumination values. Finally, the point found in block 3 in Fig. 2 is used to look up the pixel value of the currently displayed image which is then used to calculate a weighted average of the two illumination values returned from block 4.

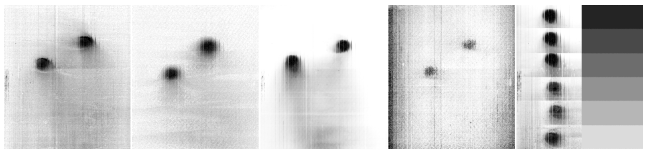
Furthermore, to stabilize the illumination value as the user interacts with the display, past illumination values are stored, and the largest stored value is used. In our implementation, illumination data is obtained at half-second intervals, and the five most-recent values are saved.

4.2 Selective normalization

The process in block 8 in Fig. 1 normalizes the calibrated input images, i_{on} and i_{off} (backlight on and off, respectively), so that zero represents full touch and the other represents no touch. Processing for white pixels in the display image is a trivial case; $1 - |i_{on} - i_{off}|$ is calculated and then scaled linearly accordingly. For black pixels, the illumination has to be considered since the backlight is absorbed by these pixels, resulting in less reflection from the finger. If the illumination is greater than a fixed parameter γ, only i_{off} is used because the difference between block and non-blocked light is high enough. Otherwise, if the illumination is weaker than γ, i_{on} is used as the backlight reflection is sufficient. The resulting image for black pixels is then corrected by increasing the contrast to discard shadows that do not represent touch. Finally, the two values for black and white pixels are averaged by using the displayed pixel value as a weight.

5. Results

The implementation operates at 60 fps without leveraging a GPU. The illumination approximation does not diverge more than approximately 15 % from the actual value when the user interacts with the device. As observed in Fig. 3, when the display image is white, clear touch regions can be distinguished regardless of illumination. The black display image is also clear in bright ambient light. The result for a black display image with very low ambient illumination presents the touch regions decently; however, the result is noisy and vignetted, and the dynamic range is lower. When illumination is close to γ, there is a *dead-zone* in black display images because the reflected light from the backlight has a value similar to the incident ambient light. Overall, compared to a method that does not consider ambient light or the displayed image, our solution performs better since it also operates when the screen is non-white.



(a) white bright (b) white dark (c) black bright (d) black dark (e)

Fig. 3 The output images from the selective normalization process. Figure (a) to (d): Results with white and black displayed image, bright and dark ambient light, as stated beneath. Figure (e): Six frames captured in bright ambient light with varying gray levels.

6. Future work

The development of a customized blob detection algorithm specific to the proposed approach is a future goal. Specifically, to find shapes in extremely noisy and vignetted images, a blob detector for black display images in dark environments needs to be developed. In addition, as suggested in Section 1, the algorithm can be extended to include more advanced features such as gestures (for touch and hovering above the screen), hand detection, and scanning capabilities.

7. Conclusion

Using an in-cell optical touchscreen capable of capturing sensor information when backlight is turned on and off, a method to obtain images with emphasized touch contact regions that can be used for touch detection was developed. To achieve this, both estimated ambient illumination and the displayed image are utilized.

References

[1] Abileah, A. and Green, P.: Optical Sensors Embedded Within AMLCD Panel: Design and Applications, *Proceedings of the 2007 Workshop on Emerging Displays Technologies: Images and Beyond: The Future of Displays and Interaction*, EDT '07, New York, NY, USA, ACM (2007).

[2] Brown, C., Hadwen, B. and Kato, H.: A 2.6 inch VGA LCD with Optical Input Function using a 1-Transistor Active-Pixel Sensor, *Solid-State Circuits Conference, 2007. ISSCC 2007. Digest of Technical Papers. IEEE International*, pp. 132–592 (2007).

[3] Buxton, B.: Multi-touch systems that I have known and loved, *Microsoft Research*, Vol. 56, pp. 1–11 (2007).

[4] Lee, D.: The state of the touch-screen panel market in 2011, *Information Display, March* (2011).

[5] Microsoft Corporation: The power of PixelSense, http://www.microsoft.com/en-us/pixelsense/pixelsense.aspx (2012). Accessed: May 14, 2014.

[6] Tamura, H., Hamada, T., Nakagawa, T., Aoki, T., Ikeda, M., Kozuma, M., Kurokawa, Y., Ikeda, T., Moriya, K., Hirakata, Y., Kamata, N., Murakawa, T., Koyama, J., Yamazaki, S., Tochibayashi, K., Okazaki, K. and Sakakura, M.: 50.2: High Reliable In-Ga-Zn-Oxide FET Based Electronic Global Shutter Sensors for In-Cell Optical Touch Screens and Image Sensors, *SID Symposium Digest of Technical Papers*, Vol. 42, No. 1, pp. 729–732 (2011).

[7] Walker, G.: Display Week 2012 Review: Touch Technology, *Information Display*, Vol. 9, p. 12 (2012).

[8] Walker, G. and Fihn, M.: LCD in-cell touch, *Information Display*, Vol. 27, pp. 8–14 (2010).

SS3-54
The 17th Meeting on Image Recognition and Understanding

Image processing for in-cell optical touchscreens

Philip IRRI
Julian LINDBLAD
Hikaru TAMURA
Jiro IMADA
Shunpei YAMAZAKI

What is an in-cell optical touchscreen?

The most common touchscreen technologies used today are based on resistive and capacitive types which detect touch input from pressure and electro-magnetic fields. In-cell optical technology on the other hand uses photodiodes embedded in the pixel-layer which enables new possibilities such as hand recognition, non-touch gestures, bar- and QR-code reading, and fingerprint scanning.

What is the challenge?

Highly dependent on:

- Surrounding ambient light
- Image displayed on screen

Can be solved using infrared backlight

- But increases power consumption

We are solving this using software!

Touchscreen specifications	
Display size	6-inch
Type	In-Ga-Zn-Oxide (IGZO) FET LCD
Manufacturer	Semiconductor Energy Laboratory Co., Ltd. (SEL)
Backlight	White LEDs
Display resolution	768 (H) x 1024 (V) (XGA)
Sensor resolution	384 (H) x 512 (V)
Sensor capture rate	60 Hz
Shutter type	Global shutter

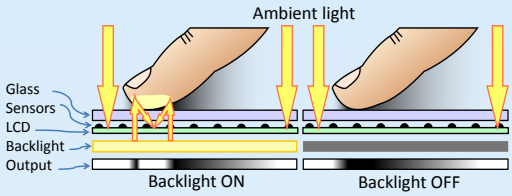


Figure 1: *Overview of the hardware. The touchscreen is able to capture two sensor images per frame; one image is taken with the backlight turned on and one with the backlight turned off.*

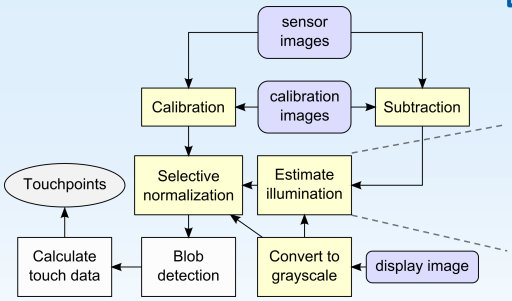


Figure 2: *Flowchart showing the steps in our proposed algorithm. The gray blocks are standard procedures and are not described here.*

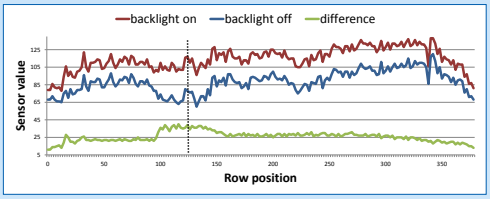


Figure 3: *Showing one row of sensor values for backlight on and off respectively, as well as their difference. Notice how the peak in the difference curve clearly suggests a touched contact area (actual point indicated by the vertical dashed line).*

Algorithm overview

The proposed algorithm considers both ambient illumination and pixel values of the on-screen displayed image when detecting and calculating touchpoints.

Ambient light estimation

The maximum sensor value is fed into two functions separately which converts it to lux, one each for black and white display images, that have been approximated on premeasured test data. The corresponding pixel value in the displayed image is then used as a weight to average these two values.

Selective normalization

A per-pixel operation on the sensor data gives us touch information for white and black display pixels:

$$white = |backlight\ on - backlight\ off|$$

$$black = \begin{cases} backlight\ on, & if\ low\ illumination \\ backlight\ off, & if\ high\ illumination \end{cases}$$

These values are normalized such that touched areas all have the same value. Finally, depending on the actual corresponding displayed image pixel value, *di* [0-1], the resulting pixel is averaged:

$$result = white \cdot di + black \cdot (1 - di)$$

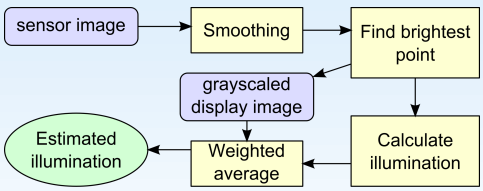


Figure 4: *Flowchart showing the illumination estimation function block.*

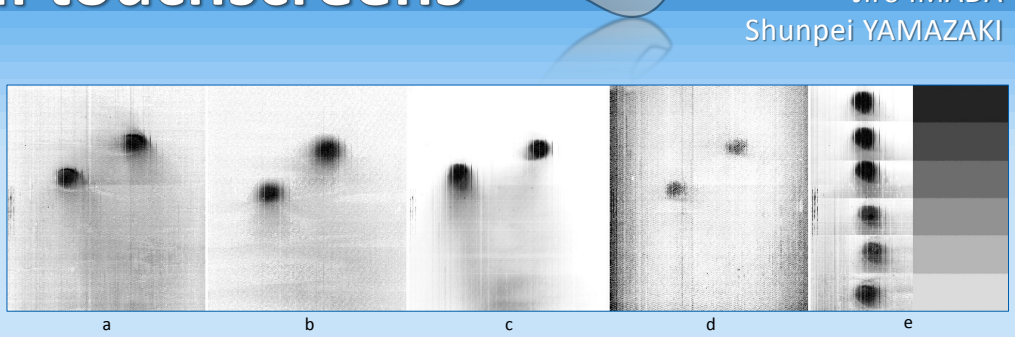


Figure 5: *Result images captured in various conditions. Note that the touch contact areas are clearly visible in all cases.* ***a)*** *White display image in a bright environment.* ***b)*** *White display image in a dark environment.* ***c)*** *Black display image in a bright environment.* ***d)*** *Black display image in a dark environment.* ***e)*** *Six touchpoints in varying blackness levels in a bright environment.*

Results

- Implementation runs at 60 fps
- Good ambient illumination estimations (Figure 6)
- Generates clear touch contact area images for various working conditions (Figure 5)
- Test with 128 touchpoints (Figure 7)
  - True touchpoints: 105 vs. 58
  - False touchpoints: 18 vs. 58

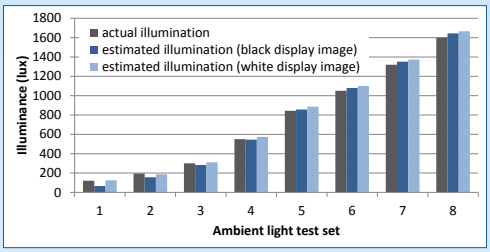


Figure 6: *Results of the ambient illumination estimation process. Both estimations for black and white display are shown. As can be observed, the estimation does not diverge significantly.*

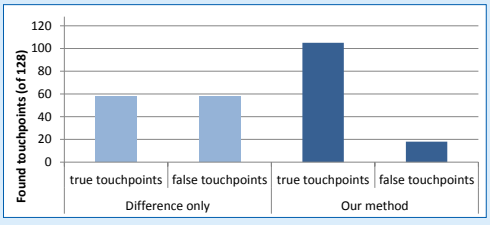


Figure 7: *Results from our algorithm compared to using only the difference between backlight on and off. This result is based on 128 touches ranging from 0 to 1600 lux on black and white display images respectively. Both tests use the same global threshold segmentation and connected components method for finding the touchpoints.*

Summary

- ✓ A method to obtain images with emphasized touch contact regions in various working conditions was developed
- ✓ Uses a calculated estimation of the ambient illumination and the currently displayed on-screen image as input
- ✓ increases the number of true touchpoints by ~80%
- ✓ decreases the number of false touchpoints by ~70%

References

- Tamura, H., Hamada, T., Nakagawa, T., Aoki, T., Ikeda, M., Kozuma, M., Kurokawa, Y., Ikeda, T., Moriya, K., Hirakata, Y., Kamata, N., Murakawa, T., Koyama, J., Yamazaki, S., Tochibayashi, K., Okazaki, K. and Sakakura, M.: 50.2: High Reliable In-Ga-Zn-Oxide FET Based Electronic Global Shutter Sensors for In-Cell Optical Touch Screens and Image Sensors, SID Symposium Digest of Technical Papers, Vol. 42, No. 1, pp. 729-732 (2011).
- Walker, G. and Fihn, M.: LCD in-cell touch, Information Display, Vol. 27, pp. 8–14 (2010).

SEL SEMICONDUCTOR ENERGY LABORATORY
CHALMERS UNIVERSITY OF TECHNOLOGY