Institute for Visualization and Interactive Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit Nr. 237

Camera-Based Finger Recognition to Improve Touchscreen Input

Stephan Roth

Course of Study:	Informatik
Examiner:	Jun.-Prof. Dr. Niels Henze
Supervisor:	Dipl.-Inf. Sven Mayer, Dipl.-Inf. Dominik Weber
Commenced:	May 5, 2015
Completed:	November 4, 2015
CR-Classification:	H.5.2

Abstract

Today's mobile devices have changed the way of interaction with computers. The touchscreen has to fulfill all requirements that prior were fulfilled by input devices like mouse and keyboard. These offer multiple levels of interaction through different mouse buttons and modifier keys. Instead, the input on touchscreens is flat. Additional dimensions like the touch duration extend the input vector but decrease input speed. Other options like movements gestures or input with multiple fingers are less accurate and require more space. To enable multiple levels of input on the touch screen we take the finger type as an additional dimension into account. We envision, that this is faster and more intuitive for users. Related work shows a wide range of possibilities in hand and gesture detection, but little methods on finger recognition. We enrich previous work by investigating a method for fully mobile prototypes. Touchscreens in today's mobile devices are not able to detect the finger types, while the finger touches the screen. Hence, we build our own prototype by combining a tablet with a camera. We iteratively detected image features and trained finger recognition. We created a model based on data we collected in an experiment. Within 20 frames our method detects up to 72% accuracy of our participants' hands and recognizes the correct finger with 70% accuracy.

Kurzfassung

Durch mobile Geräte hat sich unsere Interaktion mit Computern verändert. Vom Touchscreen wird erwartet, dass er dieselben Anforderungen wie Maus und Tastatur erfüllen kann. Diese verfügen im Gegensatz zum Touchscreen über mehrere Eingabeebenen, wie Maustasten und Steuerungstasten. Zwar können zusätzliche Eigenschaften wie die Dauer einbezogen werden, doch dies verlangsamt die Eingabe. Andere Möglichkeiten wie Bewegungsgesten und die Benutzung mehrerer Finger besitzen eine geringere Präzision und benötigen mehr Platz. Wir möchten dagegen die Eingabe erweitern indem wir den Fingertyp in Betracht ziehen. Wir vermuten, dass dies sowohl schneller als auch intuitiver nutzbar ist. In verwandten Arbeiten wird eine Vielzahl an Möglichkeiten für Hand- und Gestenerkennung präsentiert. Dagegen ist Fingererkennung weitgehend unerforscht. Um dies zu ändern, untersuchen wir eine Möglichkeit für vollständig mobile Geräte. Heutzutage sind Touchscreens nicht fähig den tippenden Finger zu bestimmen. Deshalb haben wir einen eigenen Prototyp entworfen, der aus einem Tablet und einer Kamera besteht. In einem iterativen Prozess wurden Merkmale der Finger untersucht und eine Fingererkennung trainiert. Wir haben dabei auf Daten zurückgegriffen, die wir in einer Studie gesammelt haben. Innerhalb von 20 Frames erkennt unsere Methode Hände mit 72% Genauigkeit. Auf diesen wird, mit einer Genauigkeit von 70% erkannt, um welchen Finger es sich handelt.

Contents

List of Figures

List of Tables

List of Acronyms

CPU Central Processing Unit

ID Identification Number

Leap Motion Leap Motion Sensor: an Infrared Stereo Camera

LED Light Emitting Diode

NASA-TLX NASA Task Load Index

OpenCV Open Source Computer Vision

PC Personal Computer

SDK Software Development Kit

1 Introduction

Interaction with computers changed with the increasing varieties of device types. At the beginning they were room filling machines, today they range from large server systems to small smartwatches. Improvements in rechargeable battery technologies paved the way for mobile devices. Processor efficiency increases together with better power saving modes, making it possible to use computer devices of any kind over hours without an external power source. Devices shrunk and were combined, e.g. smartphones are able to replace compact cameras, personal digital assistants and cell phones. Today's tablets become powerful enough for typical use cases of Personal Computers (PCs), like office work and web browsing.

Due to the form factor of smartphones and tablets the interaction techniques changed. In 1963 the HP 150 had one of the first touchscreens. On smartphones and tablets the touchscreen is a powerful and intuitive way of interacting. On one side it represents contents like other displays. On other side it is introduced as replacement for mouse and keyboard.

Keyboard and mouse are input devices that were sophisticated since they were invented. Today they are powerful input devices through multiple layers of input. In contrast input on touchscreens is flat. On the one hand, it is able to detect multiple touch points and their assigned movement. On the other hand, multiple touch points are difficult to handle with high accuracy. Fingers require space and therefore multiple fingers are less accurate than one. A user could perform multiple actions for one mouse pointer position. Each mouse button has a clear sense-making function that can be used with respect to the content. Touch instead offers that there is a touch and nothing more.

On the technical point of view tapping on the touchscreen is recognized as area. For simpler interface programming this value is translated into an input vector which contains entries like position, duration and pressure. A mouse has each mouse button as a possible input dimension. Instead of pressure on a touchscreen, mouse buttons are binary and can be used in combinations on the same position. On keyboards there are modifier keys like CONTROL and SHIFT. They are, like mouse buttons, binary input dimensions. Today a part of these input dimensions is the same on all devices and systems. The left mouse button stands for activating a function, the right mouse button opens a context menu. Holding the SHIFT-KEY toggles higher and lower case letters, the CONTROL-KEY in combination with another key can activate program functions directly. On today's touchscreens multiple layers of input can be achieved by additional input values like touch duration, movement, or finger count. Duration and movement take more time than tapping. Movement of fingers and finger count, often called as touch gestures, are less accurate in position. Hence, techniques like APPLE'S 3D TOUCH take the pressure as additional input dimension into account. As a result, they can distinguish between light pressure, normal pressure and strong pressure. We think that finger types are a more intuitive alternative or supplement.

Colley and Häkkilä [CH14] tested the behavior of finger depending actions. Therefore they used a Leap Motion sensor[1] (LEAP MOTION): an infrared stereo camera. In user study they conducted, a static prototype was deployed. In contrast we want to test if finger recognition on mobile prototypes is possible.

In the last few years, manufactures produced tablets with multiple processor cores. This enables higher calculation power for games, but a better power saving if the device is idle. For normal work like office work, image viewing or simple games, often only a small part of the available calculation power is needed. We see the unused resources as potential for new input methods, e.g. the LEAP MOTION consumes more energy than regular cameras because it has an integrated light source and applies imaging algorithms on its field of view.

We combined LEAP MOTION with a tablet such that the field of view contains the hand when touching the screen. In our first tests we discovered that the LEAP MOTION's hand detection method does not work in our case. Hence, we conducted a user study to collect data for training and evaluating. We asked our participants to fill in questionnaires to get feedback and insights about the workload. Based on the collected image data, we trained a hand detection. Hands are detected by weighting edges with the optical flow and filtering of periphery edges. We manually labeled hands in the train data to enable an evaluation. Based on this labels, our hand detection was improved by empirical searching for parameters. Our finger recognition is based on our results of the hand detection. We use a classifier to recognize the touching finger by the distance to known hand images.

1.1 Outline

Chapter 2 – Related Work: Related work that shares context with this work.

Chapter 3 – Data Acquisition: We describe our used prototype. We conducted a study to collected data for finger recognition training.

Chapter 4 – Data Analysis: In this chapter we present the used pipeline and detection results.

Chapter 5 – Evaluation: Evaluation of our study and possible influence on finger recognition usage by users. Also the participants' feedback is handled.

Chapter 6 – Conclusion and Future Work: Our research and results are summarized. We also show where we see potential for future work.

[1]http://leapmotion.com

2 Related Work

Today we see a diverse range of input methods for computers and mobile devices. The large amount of research done in this decade indicates there is still a wide range of open possibilities. The input vector of touchscreens is mostly limited to the 2D position. Thus the input vector can be extended to create more freedom for the user, a more intuitive interaction or simpler user interfaces. Touchscreens are established as a powerful connection between content and interaction. Usability can be improved by higher accuracy. Nevertheless, touch interaction is restricted in its possibilities. Thus additional sensors are used to improve input methods or even generate new interaction methods. One possibility is to use cameras as an explicit input sensor. This enables extending the touch input vector with finger recognition as new dimension.

We decide between detection methods and recognition methods as follows: Detection methods detect the visibility of an object and can be used to detect position and size. Recognition additionally describes the object. For example, a finger detection detects the position of a finger in an image and the recognition adds the finger type.

In this chapter we first show multiple ways of humans' natural interacting which are employed as computer interaction. In the second section we focus on existing finger recognition methods. Then we show ways how hands can be detected by the camera. If hands are known, they can be interpreted. Hand gesture recognition has similarities to finger recognition, but the finger type is not important to detect them. We present and discuss methods in the last section.

2.1 Natural Ways of Interacting

Today's interaction with mobile devices differs from the interaction with traditional PCs. The devices are smaller, lighter and work with fewer resources. Nevertheless, they are powerful computers, equipped with a wide range of input sensors. They have cameras, microphones, accelerometers, gravity sensors, gyroscopes, and rotational vector sensors. The touchscreen is established as intuitive connection between content and interaction. Moreover, mobile devices offer possibilities for new ways of interaction between computers and human.

Humans interact with a large vocabulary. Today's input possibilities for computers of all kinds are restricted to a few of them. Since humans are able to manipulate objects with their hands, the first input methods focused on them. One of the most natural interactions is pointing at an object in environment. Nowadays mobile devices have enough resources to use this information as input, but the hardware is often not able to gather the necessary data. If the user's requirements cannot be achieved by the internal sensors, then there is a wide range of possibilities how devices can be extended. External cameras can be used and placed where they are needed. Internal sensors can be modified, for example the field of view of frontal cameras. If prototypes like these convince stakeholders, future devices may contain their sensors as internal sensors.

Yang et al. [Yan+13] used a smartphone with frontal camera to enable periphery based actions. The field of frontal cameras is designed to contain the users head if the user looks at the screen. This feature can be used for video calls or photographing oneself. To achieve their goal, they used a special lens to extend the field of view to surround-seeing. Periphery based actions can be triggered by finger pointing of the user, the user's movements, and fingers or touch pens over the screen. Interaction is possible without holding or touching the device. Also indirect actions, like leaving the room, can be used. This lens has the disadvantage of heavily distorting the field of view. While the distortion can be undone, the resulting image quality depends on the cameras resolution. The lens Yang et al. used was large and reduces users comfort in carrying such a device.

Gestures are natural for humans. We use them often while communicating to make things clear. Cutler and Turk [CT98] enabled a static computer to detect and interpret body movements like flapping or clapping. Therefore, they used a camera which covers a large area with its field of view. Body position and movement were approximated by an optical flow method. To simplify the conditions, they reduced the image data by taking only skin colored pixels into account. For this, hands, head and other skinned parts are separated from background and also a hand detection can be applied. To enable real-time applications, the optical flow is approximated.

If a device is held, the motion becomes a possible input method. Tapping at different positions on a mobile device results in typically motion events. McGrath and Li [ML14] used an accelerometer to detect these events. To increase the difference between their touch events, they focused on five different positions at the device's sides. Their method is usable like additional buttons on the device. Moreover, it is more flexible than hardware buttons, because it needs less counter pressure. Tap positions can be configured. Furthermore it is possible to deactivate them if the user does not need them, for example if she turns her device in standby mode.

2.2 Interaction with Touchscreens

The touchscreen combines digital content and physical input. Thus it is intuitively usable by hand or pen. However, touchscreens do not recognize the exact region, where users want to tab. This can be confusing or frustrating for users. Holz and Baudisch [HB11] studied touch input and described error causes. Users try to point their targets, but angle of view and the fingers shape and roll trigger another position. Touch positions are not as exact as the position of a mouse cursor. Touch input can be understood as pointing of the user at a specific point on the screen. Caused by the finger shape and finger rotation the recognized touch positions differ up to 4 mm. The head position and therefore the angle of view also influence how users tap and therefore the error size. Holz and Baudisch improved the touch accuracy to 1.6 mm with a little spot light that shows the touch position on the finger nail.

Causes for targeting errors on touchscreens, like fingers shape and head positions, are undetectable by touchscreens itself. They only detect the error prone contact area of the finger. However, it is possible to support the user by shifting the touch position before handling. Buschek and Alt [BA15] corrected the touch position with a machine learning algorithm. In a study they asked participants to tap at touch targets and used the measured error to calculate a position depending correction matrix. While this was device specific, Buschek et al. [BRMS13] also studied the creation of user based touch correction with normalized correction matrices that fit any device. Both solutions do not take finger rotation and head position into account.

Correcting the input improves usability and can additionally be used to shrink button sizes to win content space. Usability can also be improved by a wider range of input dimensions. Further techniques, like long press, take another dimension of the input vector into account, like the time. Gestures use the movement and finger count, and if there is no fitting dimension a new one can be created, like the pressure in APPLE'S 3D TOUCH.

Touchscreens in today's mobile devices are not able to detect more than the touch region. They cannot detect if a finger, another skinned part or a touch pen touches the screen. Certainly it can be made more sensitive to detect finger over the touchscreen. Rogers et al. [Rog+11] showed that with existing touch hardware it is possible to predict the orientation of the touching finger. This enables a context based correction of the touch position. It also makes new interaction concepts possible, like their high-precision bimanual pointing where the pointing directions of two touching fingers are taken into account.

Touchscreens are frequently used by finger, but there is a difference between individual finger types in size and comfort. The thumb for example has fewer degrees of freedom compared to other fingers. The index finger is more comfortable to use than the little finger. To proof this Colley and Häkkilä [CH14] tested the topic in multiple user studies and with a stationary prototype which contains a LEAP MOTION. Their results show that fingers differ significantly in performance and comfort. The subjective rating of their participants showed that the index finger is the most comfortable and fastest finger. With increasing distance to the index finger, speed and performance decreases. So the little finger takes the lowest rating. The thumb has a large variance in comfort, but value based it stays between middle finger and ring finger. Index finger and middle finger are the most accurate. In a second study they measured the error per finger at touch screen positions. Their result showed that hitting targets in the corners is more difficult than in the middle. Especially hitting the upper left corner with ring finger or middle finger is very hard. In their third study the concept of finger specific functions was tested. In Figure 2.1 their functional prototype with a smartphone and a LEAP MOTION can be seen. The

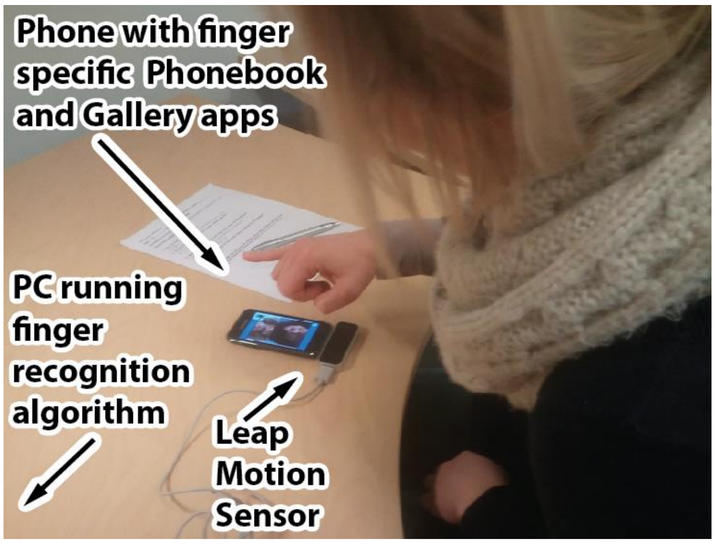

Phone with finger specific Phonebook and Gallery apps
PC running finger recognition algorithm
Leap Motion Sensor

Figure 2.1: Prototype by Colley and Häkkilä [CH14] for exploring finger specific input.

LEAP MOTION has integrated hand detection algorithms. On the smartphone they run two sample applications with per finger functions. Their participants' response was positive for functionality. Negative response mainly focused on comfort of different fingers. Uncomfortable fingers are suitable for position independent functions or critical functions like deletion.

2.3 Existing Finger Recognition

In this work we distinguish between detection and recognition as follows: A detection finds position and shape of an object. All touchscreens are able to detect fingers on their surface. A recognition additionally detects the object itself and describes it. Today's mobile devices are not able to determine the type of a touching finger. Colley and Häkkilä [CH14] used LEAP MOTION, an infrared stereo camera with an integrated light source. It is shipped with detection methods for hands, arms and fingers. Also tools like a pen can be detected. Nevertheless, finger recognition is studied on touch tables.

On touch tables other detection methods for hands and fingers than on mobile devices can be deployed. The device is static and larger in its size. Seamless presentation of content and touch detection has to handle different challenges. The touch technique used by mobile devices is not scalable. Hence, there are other finger detection methods. Chen et al. [Che+11] utilized an infrared camera to detect fingers on the tables surface. This technique is not able to distinguish between fingers and therefore is a finger detection.

Marquardt et al. [Mar+11] created a tagged glove to interact with a touch table. Hand and finger recognition is more simple if the fingers are marked. On one side, this way of finger recognition is reduced to tag detection which is more design depending and less error prone. On other side, it is only possible if users accept the tags on their hands. Marquardt uses a glove which on the one hand is uncomfortable in warm areas. As an advantage the tags can removed from the hand any time without destroying it and can be utilized by any other person. Furthermore this technique can be used on other touch devices, if there is a suitable tag detector.

Finger recognition by position is possible but we only mention it for the sake of completeness. Azenkot et al. [Aze+12] used the finger position to detect which finger has touched the screen. They did not use any visible content and therefore their technique can also be done with any other touch sensitive surface. Blind typing is possible with this technique. Today's touchscreens are not suitable for real finger recognition.

2.4 Hand Detection

Already mentioned in Section 2.1 are camera-based methods that are able to detect hands. We also introduced the Leap Motion that was utilized by Colley and Häkkilä [CH14]. In this section we present different ways how hands can be detected. Kang et al. [KNR08] presented different models for hand and finger detection for cameras. The first group of models approximates the hand position and gesture as a hand model. A 3D-description of the hands surface is called volumetric model. To derive the hand gesture and positions the model has to be varied until it fits with the camera image. On one hand these models require many parameters and are too complex to be calculated in real time. On other hand they represent hands fairly realistic. Another hand representation is the skeleton model. A virtual skeleton that fits human hands and contains its physical possibilities is used to approximate a hand on an image. The Leap Motion also describes hands as a skeleton model. The second group of models is appearance based models that only take visible features into account. The presented variety contains deformable templates, hand image sequences and shape models. These models also show that hand detections work hand in hand with finger detection methods.

Skin color is often used for hand detection. Its robustness depends on the environment light. Darkness or colored light confuses these methods. Also skin colored objects except the hand may lead to errors. Kölsch and Turk [KT04a] created a fast hand tracking algorithm for image sequences. Hands are tracked by their two dimensional position in the image. This hand detection is color-based and uses multiple features to detect the hands position. Positive features are connected to flocks of features. For their participants wore long-sleeved clothes, the center of these flocks lays over the hand. In further images, feature positions can be reused if they stay positive. If a featured position got an unsuspected color value, it is deleted. New features are created from skin colored pixels to keep the hand tracked.

If the hands shape stays roughly constant it is a possible feature for tracking. The condensation method of Isard and Blake [IB96] is able to follow a hand in an image sequence. The condensation algorithm is a shape model and works with complex background. Their algorithm was able to run in real-time with 25 Hz on a computer with 200 MHz CPU frequency. Thus, today's mobile devices are able to run this algorithm. This method is not restricted to hands but can detect other objects like faces. We assume that changes in the shape, like clenching one's fist, may confuse this method.

More robustness to background can be achieved with depth cameras. Ren et al. [RYZ11] used a Microsoft Kinect to detect hands in images. They identified fingers at detected hands by iterating over the hands contour. While they only used it to distinguish between gestures, it may be possible to extend their method to finger recognition, if the hand is in midair. Through our search we found multiple papers which focus on hand gesture recognition without hand detection. These papers are discussed in the following section.

2.5 Hand Gesture Recognition

Humans use different ways of communicating. One of the natural possibilities is speech. Unfortunately, not everyone is able to speak. Dumb people often use gestures to communicate with others but not everyone is able to understand them. Nowadays researchers develop methods to enable computers to recognize hand gestures and translate them to speech. Also controlling devices through camera-based gesture recognition is possible and used in games.

Edges are important image features. Ravikiran et al. [Rav+09] used the canny edge detector [Can86] to interpret hands in images. They derived fingertip positions from the bend of image edges. In relation to the whole hand gestures are derived. Their focus lays on sign language recognition and therefore in detection of specific hand gestures in midair. Although each gesture is a unique combination of hand rotation and stretching of different fingers, the fingers types are not important.

The goal of Kang et al. [KNR08] was enabling sign based input. Therefore, they take images from hands in midair with a flat background. Skin colored blobs are separated and morphological operations are applied. Fingers are separated from the hand and used to identify all hand depending blobs. As last step the collected information is used as input. The evaluated their method by enabling it as input for a calculator. The disadvantage of this method is the dependence on environment light. Darkness or colored light confuses the detection.

If conditions ensure that the image only contains the hand and that hand gestures are unique, hand gestures are recognizable by image frequencies as shown by Kölsch and Turk [KT04b]. They used cropped and normalized hands for their detection and compared the similarity to an artifact-free Fourier transformation of the mean of hand images with specific unambiguous gestures. We assume that the image frequencies cannot be used for finger recognition. In the case of Kölsch and Turk there was a large difference between the hand gestures. But there are nearly infinity possible hand gestures for one specific tapping finger. Moreover similar gestures for different fingers are possible.

Machine learning algorithms are also able to classify hand gestures. Mackie and McCane [MM04] used a decision tree to distinguish generated hand postures in images. Their images did not contain noise, color of background and hand were flat. However, machine learning algorithms are powerful and able to determine relevant features that work on real data.

The previously described methods focused on detection of a known set of hand gestures. However, the hand posture itself can be interpreted. Lee and Lee [LL11] searched hands in images. Stretched fingers are detected, fingertip position approximated together with the fingers direction. This method can be used as mouse replacement. They used finger movement as drawing actions and swinging in and out a finger as tapping.

2.6 Summary and Discussion

There are multiple ways of interaction with today's mobile devices. Methods are based on humans' natural interaction methods like speech and using the hands to manipulate the world. The touchscreen connects the presentation of virtual objects with physical actions. However, today's touchscreens are limited in their detection accuracy and dimension of the input vector.

Touchscreens are not able to detect what kind of objects touch the screen. Fingers and touch pens appear as touch areas and are translated into touch points. This causes shifts between target and detected touch point. Holz and Baudisch [HB11] addressed multiple error causes like the finger shape, angle and the users angle of view. Shifting matrices can be used to shrink the erroneous shift [BA15; BRMS13].

To improve touchscreen input new input dimension can be created. APPLE used pressure in its 3D TOUCH technology. Rogers et al. [Rog+11] technique ANGLEPOSE enabled detection of the finger orientation with existing touch hardware. Colley and Häkkilä [CH14] conducted user studies with a static prototype to test behavior of finger recognition as input dimension.

There is no deeper research in finger recognition as far as we know. The simplest possibility is marking the fingers with tags and detect fingers indirectly. Efficiency of this method depends on design of these tags. Marquardt et al. [Mar+11] used a tagged glove to enable finger recognition for touch tables. There are also sensors like the LEAP MOTION, where the manufacturer offers hand detection methods. In case of the LEAP MOTION there is public accessible information which method they use, as far as we know.

Hand detection methods exist in multiple variations. We showed hand detections that depend on skin color [KT04a; LL11], optical flow [CT98], shape [IB96] or depth images [RYZ11]. Skin color is usable, if there is color information and the light conditions satisfy the requirements. In static scenes the motion of hands is enough to track them. On mobile prototypes the background motion has to be filtered. Shape models are robust to complex background, but they need information about the shape. Depth images allow a separation between hands in midair. Detecting hands which touch a screen may be more difficult, because the screen has to be separated from the hand.

Often hand detection is the first step for gesture recognition, whereby gestures are special hand postures. For this hands and their fingers are detected and interpreted. This can be done by analyzing edges [Rav+09] or applying morphological operations on skin colored image parts [KNR08; LL11]. Detected hand postures can be classified by decision trees [MM04]. Additionally, if the hand detection accuracy is high enough, image frequencies can also be used as feature [KT04b]. We distinguish finger recognition from gesture recognition and finger detection because the finger type is important for finger recognition but not for both others.

In this work we address the open question how we can recognize the finger type while using touchscreen input. To do so we built a working mobile prototype and conducted a study to evaluate and analyze it. The next chapter fully describes the prototype in detail, the study and our achieved data set.

3 Data Acquisition

Our goal is finger recognition on a mobile prototype. From related work we know that camera based hand detection is possible. Finger recognition on the contrary is restricted. The manufacturers of LEAP MOTION studied this method, but only for a few cases. However, hand gestures are well understood.

Colley and Häkkilä [CH14] tested usability of finger recognition in multiple user studies. For this they used a LEAP MOTION in a stationary prototype. Their results show that finger detection is a powerful way of interacting. Our work focuses on enabling mobility to users. Therefore, we build our own prototype which is described in the first section of this chapter. Using this prototype we conducted a user study to collect data to train our own finger recognition method. Design, participants and procedure of our user study are described in the middle of this chapter. Furthermore our received data set and its properties are presented. At the end we summarize and discuss the content of this chapter.

3.1 Apparatus

Our goal is to extend the input vector of touchscreens on mobile devices. For realizing the finger recognition we connected a tablet device and a LEAP MOTION which is an infrared stereo camera. We decided to use a LEAP MOTION sensor because we know it from related work [CH14]. The LEAP MOTION is developed for hand recognition. It comes with Software Development Kit (SDK)[1], containing a hand, finger and tool recognition functions. Recognized hands can directly be accessed by using their virtual skeleton model. It is not guaranteed that this hand detection works in all situations because the manufacturer of the LEAP MOTION made assumptions. They expect the hand in a distance of at least 7 cm. Also, they assume that the users lay the LEAP MOTION in front of them on a table or mount it on a OKULUS DK2[2]. Therefore, they assume to see the whole hand with fingers from below or above.

Like Colley and Häkkilä [CH14] did, a way of building a working prototype with hand recognition ability is to lay the LEAP MOTION under the touchscreen, as visible in Figure 2.1. Also, the LEAP MOTION can be mounted above the screen such that the user can work below it.

[1]http://developer.leapmotion.com
[2]http://leapmotion.com/product/vr

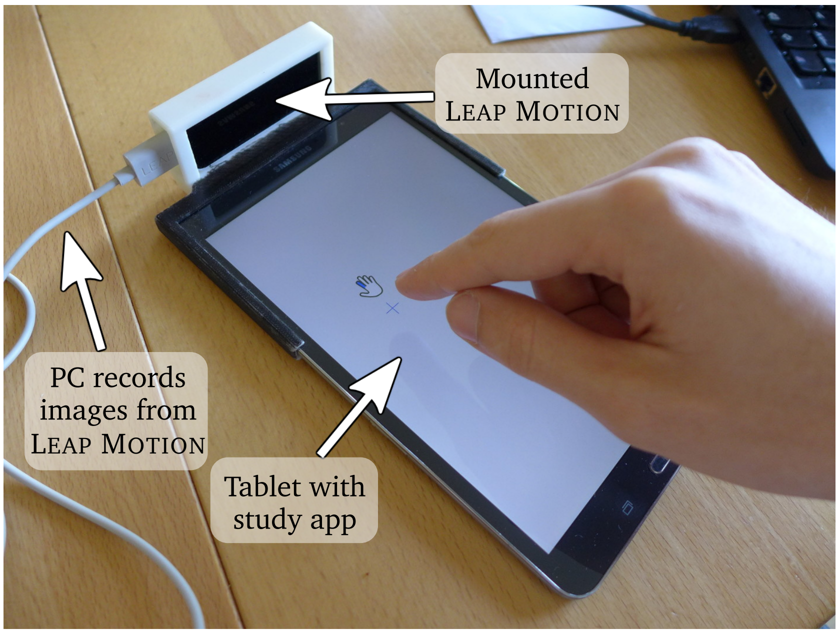


Figure 3.1: Our prototype for data acquisition in the user study. Showing the tablet, the LEAP MOTION and its mounting. The distance between the displayed hand symbol and the LEAP MOTION amounts to 8 cm.

Such a prototype is not comfortable for daily use, because it is too big. Instead we mount the LEAP MOTION at the top of our tablet, as visible in Figure 3.1. The top of the tablet screen stays closer than 7 cm from the LEAP MOTION. Thus, we win more compactness of the prototype. Middle and lower parts of the tablets front face are in the interaction range given by the manufacturer. In Figure 3.2 an example frame of the resulting field of view is shown. It covers the whole area over the touchscreen, but there is a large difference between both views. The LEAP MOTIONs aperture angle of 150 degree ensures that we can observe tapping hands at any position on the screen. We decided to use only the portrait orientation of the tablet to minimize lens distortion in the upper corners. If we would use the horizontal orientation of the tablet then the LEAP MOTION requires a distance to the tablet to cover these corners. As a result of our mounting the tablet is a static object in the field of view while user and background are allowed to move.

When testing the hand detection algorithms of the LEAP MOTION these were not able to detect touching hands on out prototype. According to the LEAP MOTIONs manufacturer[3], they use algorithms to remove ambient light and background objects like the head. Only to objects which are probably hands or tools, a 3D-reconstruction is applied and used to filter hands and

[3]http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/

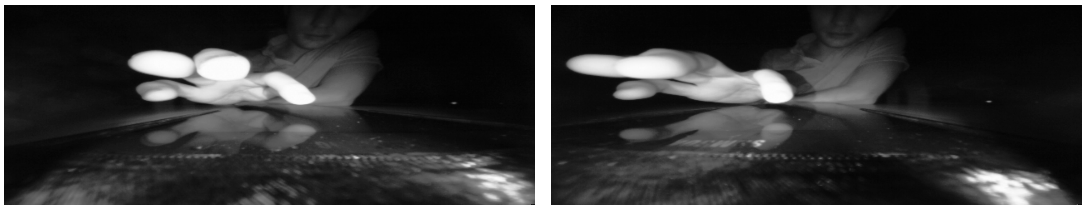

(a) Left image (b) Right image

Figure 3.2: Our prototype's field of view. The lower quarters of both images show the mounting of the LEAP MOTION at the upper side of the tablet. The tablets screen is visible in the middle where mirror images appear. Screen content is not visible, because it contains no infrared light.

tools. Since the tablets screen is close to the LEAP MOTION, this is also true for the hands. Thus, we assume two main factors that disable the hand detection. First, the tablet which is too bright for a background object. Second, the lens distortion has too much influence on touching hands and disables an accurate hand reconstruction. The latter in combination with perspective effects covering of fingers by other fingers or palm of the hand. This effect is visible in Figure 3.2 where the thumb is hidden behind the index finger. It is possible that fingers are visible in only one of both images. Hand posture and rotation can be important factors. Nevertheless, hands are bright objects with clear edges if touching the screen. This is why we trained our own hand detection and finger recognition.

The LEAP MOTION SDK interface is compatible to fourteen programming languages and program libraries, like Java and Python which we are using. The SDK uses the observer pattern to inform about new available frames. Depending on user settings the LEAP MOTION can offer up to 120 frames per second according to the manufacturer. Each frame contains two grey-level images with a resolution of 640×240 pixels. The LEAP MOTION has infrared LEDs inside to light the scene in front of the cameras. Thus hands over the screen appear as bright objects in the field of view.

As a tablet device we utilize a SAMSUNG GALAXY TAB 8.4 PRO. It supports the current alpha version of the LEAP MOTIONS ANDROID SDK due to the powerful CPU and the 2 gigabyte of memory. Open Source Computer Vision Library[4] [Bra00] (OPENCV), a computer vision library, is available for the tablets system Android. In this way we built a base for future work. On the hardware side the tablets screen does not emit infrared light and is therefore black on all images of the LEAP MOTION as visible in Figure 3.2. We used a black version of the tablet, because of the LEAP MOTION's infrared light source. A bright device surface reflects the infrared light except on the content area of the screen. While testing our prototype with a

[4]http://opencv.org

bright mounting and a white tablet we found that this creates sharp edges and bright regions. These regions were hard to distinguish from the hand. In addition, the resulting images were always overexposed.

For data collection we built a simple Java program to record the provided images in real time. The tablets memory and storage space are restricted. So we used another device, a laptop, for recording. To reduce calculation effort during the study we do not apply any imaging algorithm and save each frame with a time stamp as binary stream. This is necessary because storing the images needs time, while our finger recognition has no need to reuse images. This requires a conversion of the saved files to train our algorithms, but this can be made in one step.

3.2 Design

The study focuses on collecting data to train and evaluate hand detection and finger recognition challenges. Therefore, a repeated measure design was used to create comparable data. In this case data means the image data from the Leap Motion which were recorded by a laptop, and all data about our participants and actual study state which were collected by the tablet in our prototype. The tablet collected sensor data from accelerator, gyroscope and magnetic field sensor. Additionally, it saved all touch events together with target and touch position. All of this data contains also a time stamp. We synchronized clocks of these devices over network connection for easier combination of the data. Touch targets where given as dependent variables. The sensor data were used to decide if the participants held the tablet in the left hand or if it laid on the table during the study. This knowledge was used as independent variable.

We used a desktop PC as extra device for questionnaires. We asked each participant about age, gender, profession, experience with touchscreens, and if they are right-hander or left-hander. During the study we asked the participants to fill in twelve raw NASA Task Load Index [Har06] (NASA-TLX) one after each task or subtask. At the end we asked them for feedback about the study and about ideas, where finger detection can make sense. We also asked them to rate if they would use it or not.

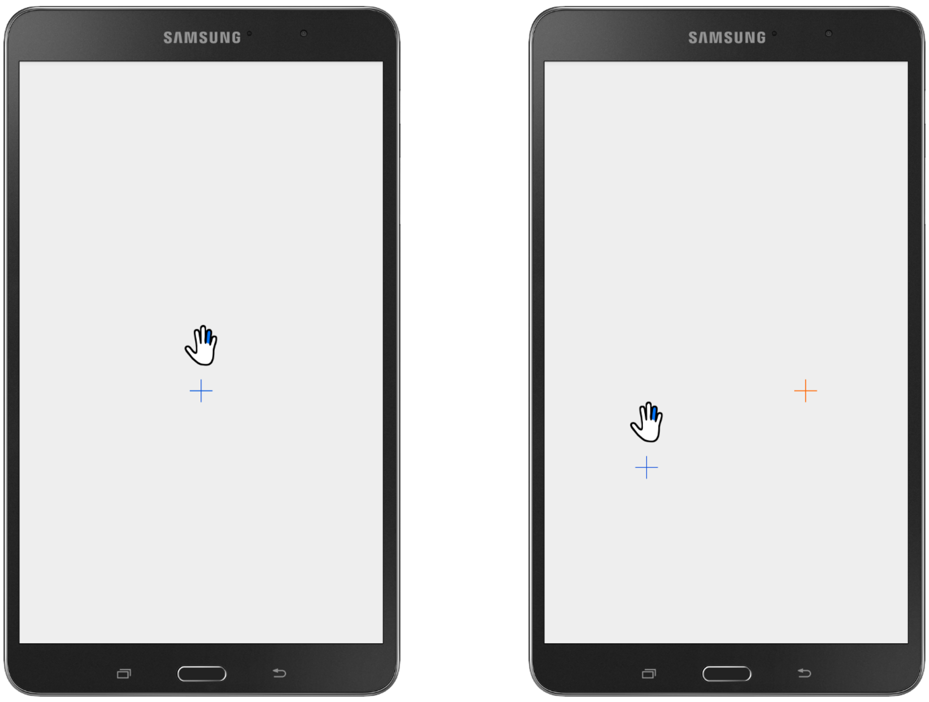
The study was structured as follows:

Task 1: 40 times tabbing at given positions for each finger in 5 subtasks

Task 2: 80 times tabbing at given positions with given finger

Task 3: 40 times line drawing with given start and end point for each finger

Task 4: 80 times line drawing with given start and end point with given finger in 5 subtasks



(a) Tabbing in task 1 and 2.

(b) Line drawing in task 3 and 4.

Figure 3.3: Examples of the study screens. The blue cross hair is the tabbing target (a) or starting point (b). The red cross hair shows the end point (b). The hand symbol indicates the finger for tapping or line drawing. In this case it is the ring finger.

Our study application interface was simple, as visible in Figure 3.3. In task 1, shown on the left side, we used a grid of 5×8 targets in randomized order for each finger. Also the order of fingers was randomized to prevent side effects. After each condition we asked to answer a raw NASA-TLX to get insights if there are any fatigue effects during the study.

Task 2 was similar to task 1. We additionally changed the stated finger between the positions. Each finger was used 16 times in randomized order. We designed the second task to have deeper insights on hand posture, switching fingers during interaction could lead to different postures. So we asked the participants to change the finger, to see if the hand gestures will be more different.

Tasks 3 and 4 were similar to the ones before. An example picture of these tasks is shown in Figure 3.3 (b). In addition to the tasks before we added red crosshairs on the same grid point positions with an offset to the blue one. In these tasks, our participants were asked to draw a line from the blue crosshair to the red crosshair. We did not give them an explicit line to follow as this is an analogy to drag and drop operations where the way is not important.

3.3 Participants

We recruited 22 participants (4 female) through our university mailing list. All of them were students of computer science or software engineering. They all reported good or very good touch screen experiences. Average age of the participants was 22.15 years (SD = 3.22) with a range between 19 and 31 years. Three of them were left handed, but also used the right hand in our tasks.

3.4 Procedure

First we welcomed our participants and informed them about the procedure of the study. Second we asked them to fill in a consent form and a questionnaire with personal data. We explained the study app and our first task. The participants were informed by a special screen with light blue background color and a short dialog when they fulfilled a task or subtask. After they fulfilled the first subtask of task 1, we showed them the first NASA-TLX. The NASA-TLX is a questionnaire designed to record the workload of a task. We asked them to fill it in spontaneous to reduce stretching of the truth. After each of the following subtasks and tasks we asked them to fill in another NASA-TLX. During the study we guided them and informed them about the requirements of each task and answered questions about the study. After the last NASA-TLX we asked them about study feedback and possible usage of finger detection. It was also written in a questionnaire. When our participants left, we served each of them chocolate as a reward.

3.5 Data Set

Our collected data contains video files, taken with the Leap Motion sensor and data about the study itself, like indicated finger, positions of fingers and targets. We received about 5 hours and 50 minutes of uncompressed video files with an average frame rate of 110 frames per second. Each video frame contains two images with a resolution of 640×240 pixels. For each participant we got 280 tapping scenes and 280 line-drawing scenes. There are situations within the video files where no finger is visible on our screen. This happened when participants took time for searching the next cross or if they filled out a raw NASA-TLX in task 1 and 3.

We recorded 12,320 touch events, from touch down to touch up, together with study data, like indicated finger, cross positions, and touch positions. Thus, we are able to detect wrong finger positions and accidental loss of contact with the screen. The data also contains confusion errors where participants used the wrong finger. For the first two tasks we corrected the data while training the finger recognition in Section 4.2 and got an error rate of 1.8% (95 taps).

Captured sensor data from magnetic field sensor, accelerator and gyroscope amounted to 556.4 MB. With their help we identified 6 participants who held the tablet in their hand while other 15 laid it on the table. The last group contained all three left-handers. Although they used their right hand we put them in a separate set to prevent any side effects. In Chapter 5 we discuss study results which are independent from our aim to collect data for algorithm training.

3.6 Summary and Discussion

In this chapter we introduced our prototype which we used to collect data in our study. We connected a LEAP MOTION sensor which is an infrared stereo camera with a tablet. To ensure the field of view contains the whole area over the screen, we mounting was installed at the tablets top. The LEAP MOTION manufacturers hand detection algorithms are unable to detect touching hands, caused by our mounting position and the resulting distortion.

Our study was structured into four tasks with a questionnaire about personal data at the beginning and another questionnaire about feedback at the end. In task 1 and 2 we asked our participants to tap with an indicated finger on cross hair targets on the screen. While in task 1 each finger was used in its own subtask, in task 2 we randomly switched between fingers. After each subtask in task 1 and after task 2 we asked our participants to fill in a NASA-TLX which is a questionnaire that is designed to capture information about workload. Task 3 and 4 were repetitions of task 1 and task 2, except that tapping was replaced by drawing a line from a start cross hair to an end cross hair.

We collected video files for each task with the LEAP MOTION sensor. The tablet collected all study data, like targets, indicated fingers, any touch event, and the device motion. With the NASA-TLX we reported subjective workload ratings of each finger, each task, and the whole study. Additionally, we recorded feedback about the study and ideas of finger recognition use cases from our participants.

4 Data Analysis

In this chapter we analyze the collected image data from our data acquisition study. Our focus lays on detecting one single finger of the right hand at a touching moment. For this we built a pipeline which is shown in Figure 3.2.

In the first section we will describe our hand detection. At the beginning we describe the data we use for training and how we extract them from our collected data. From related work we know that gesture recognition with edge detection is possible on images with flat background [Rav+09]. We also know that optical flow can be used to find moving hands in an image [CT98]. Our hand detection is based on the assumption that a hand has to move before the screen is touched. Therefore, a combination of optical flow and edge detection is used. All parts of the hand detection are described and brought together. Furthermore, we describe how we chose parameters and how our evaluation process works.

The second section concentrates on finger recognition. It is trained on hand images, extracted by results of our hand detection. For our finger detection we have to handle factors like lens distortion, hidden fingers, and different hand gestures per finger. It is also possible that there are similar gestures for different fingers. To simplify this challenge we assume that there are similarities between hands with the same touching finger type.

4.1 Hand Detection

The LEAP MOTION offers only gray-level images which stand for the infrared channel. In consequence we cannot use any skin color detection, but human skin is a good infrared light reflector as visible in Figure 3.2. The LEAP MOTION uses three cone shaped light beams to ensure a minimum brightness of the illuminated hands. As a result, at the image borders we have to consider a loss of the hand brightness.

As we want to create a new input dimension, we want to keep the reaction time as short as possible. Therefore, we have to recognize the specific finger in the moment it touches the screen. Since hand detection is our first step, we train hand detection to find the hand before the touch event is recognized by the touch screen.

If the user is going to tap we assume that at least one finger has to move. Also hand detection by optical flow is possible [CT98] but not efficient enough for fast hand detection if calculated

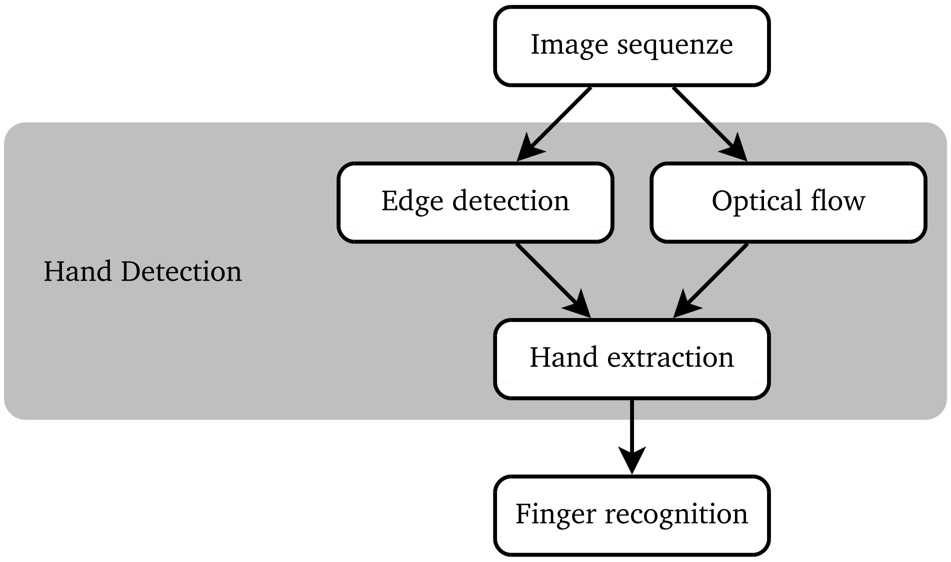


Figure 4.1: The finger recognition pipeline.

with high accuracy. On the hands in our images there are less good-to-track-features. Thus, we are using Farnebäcks two image based dense optical flow variant [Far03] which is implemented in OPENCV. Between these images movement of any visible object is approximated. There are also feature based optical flow methods but the hands in our images did not show enough features to be tracked.

To develop our hand detection algorithm we extracted images from the first two tasks of all participants. We used the captured touch down events to extract series of 20 frames out of the video files, whereby the last frame shows the moment when the finger touches the screen. Therefore, we manually shifted the video files with a time offset to remove latency of the touch screen. The latency was about 10 ms, due to the frame rate of 110 frames per second we cannot approximate it with a higher accuracy. Each of our extracted frame series comes with time stamps. Together with all other captured time series we build a data base. The time distances between two time stamps in this data have a big variance. Our analysis is image based, therefore we interpolated values of the other time series to fit the image time stamps. We used linear interpolation to fill the gaps.

Each entry in our database contains the following data: The frame series, containing left and right images per frame, ID of the participant, indicated finger and aim position, touch position and sensor data. For evaluation we inserted the hand region for each image at touch time. We manually added them as a rectangle, approximating a perfect hand detection result. The rectangle contains as much of the fingers as visible, but as less periphery as needed. In the following we call them hand labels.

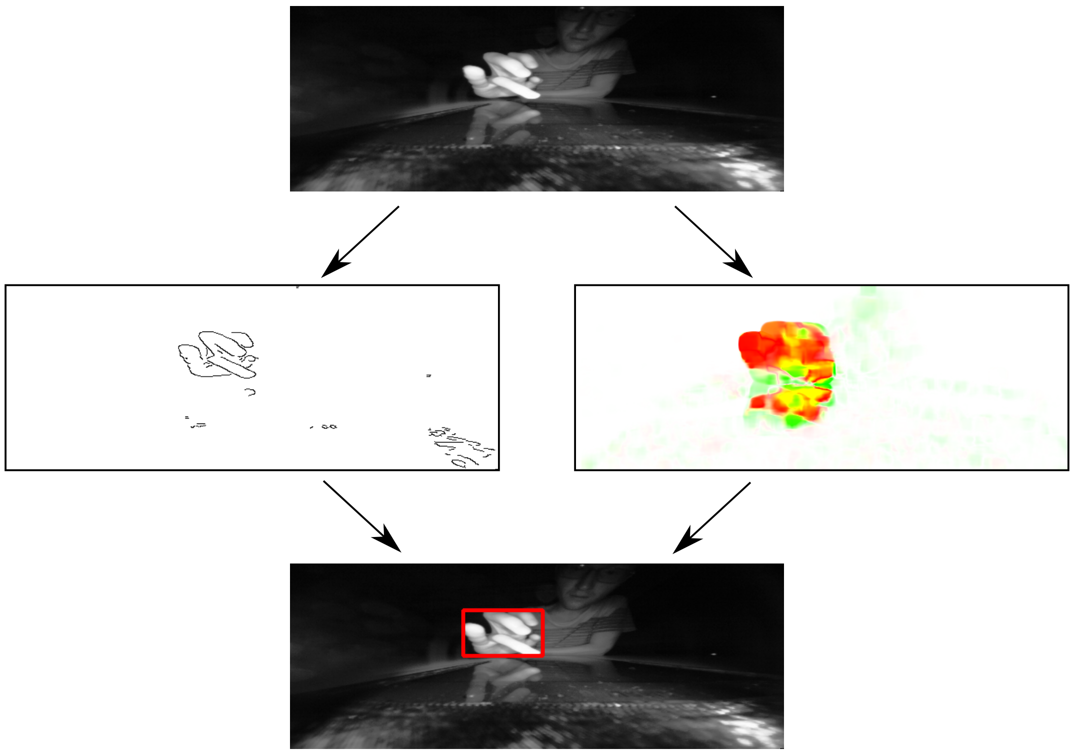


Figure 4.2: Hand detection pipeline as image flow. Arrangement according to Figure 4.1. Edge detection (left) and optical flow (right) are calculated and used for hand detection (bottom).

For hand detection we used the grey deposited region in our pipeline model visible in Figure 4.1. In Figure 4.2 example images of each of the steps are shown. At the beginning of this pipeline we approximate the optical flow for the whole image, shown on the right in Figures 4.1 and 4.2. The most important part of the result will be the down movement which is presented with the red color channel. This gives us an approximated hand region with noise. In bright regions like near fingers cups the LEAP MOTION crops values to the maximum value. In affected regions approximation of zero movement is a known issue. At the beginning we selected parameters for the optical flow by visual examination. Also the time distance of the selected images is a parameter.

We improve our hand detection with the canny edge detector [Can86] shown on the left in Figure 4.2. It guarantees that each edge has a thickness of exactly one pixel. It depends on two parameters an upper and a lower threshold. The edges taken into account are influenced by the upper threshold. The length of these edges is influenced by the lower threshold. Edges are indicated by an image where the upper threshold is applied. Only these edges are taken from an image where the lower threshold is applied. As final step the thickness of the edges is set to one pixel.

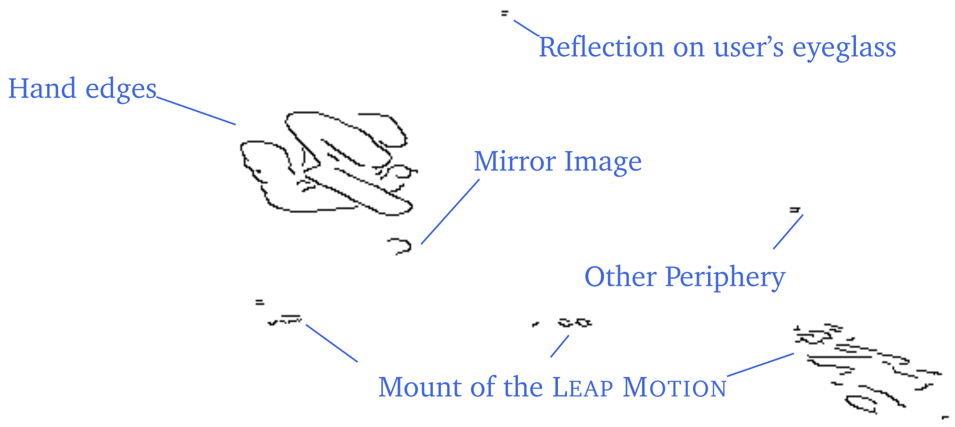


Figure 4.3: Example for edges in the LEAP MOTION's image together with causes.

We chose the parameters of the canny edge detector to detect only contrast rich edges. In Figure 4.3 we present one example of an edge image. Calculated edges are weighted with the result of the optical flow and are post filtered. The movement of an edge is approximated as the sum over the down movement in its bounding box. This eliminates static edges like edges on the prototype itself. Positional filtering is also applied. Edges in the upper regions are in the peripheral or part of hands that fill nearly the whole image when touching. In latter case this edges will be long. Our assumption is that edges of the periphery will end far away from the visible screen and therefore we can ignore edges where the lowest point is far above. We also use positional filtering to reduce edges of the mirror image. Here our assumption is similar. Edges of the mirror image can only appear on the visible screen and therefore start in this region. Here we filter according to the highest pixel of an edge. We first approximated values for these filter methods and refined them empirical.

The loss of brightness at the sides of the image results in a cut of the edges. This phenomenon also appears at the top of the image. To reduce it we enlarge of contrast and increase brightness in these regions. We are using a pixel-wise static factor which is multiplied onto each frame before edge detection. The factor map contains float values larger or equal to one. Towards the edges the value increases. For our prototype we are using simple rectangles near the edges together with Gaussian blur to create a soft blending.

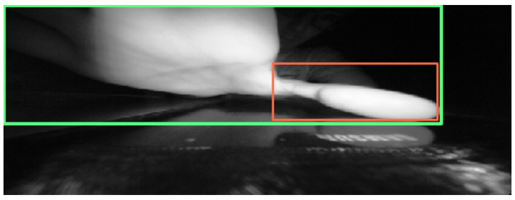
A value for the movement set to 0.05 seemed to work reasonable good in our first try. The upper border for our filtering was set to 40 pixels because this removed perfectly the reflections caused by eyeglasses worn by some of our participants. The screen appears between the 120th and the 152nd pixel row. Therefore, our lower border is set to 120 pixels. Additionally, we can crop the lower part of the image completely for it never contains touching fingers or parts of the hand. With respect to the finger size we crop only away the image parts under row 180. This value is also refined in empirical way. In the same way we choose the size of the rectangles in our mask image, its value, and parameters of the Gaussian blur.

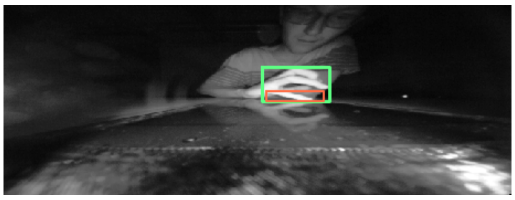
The last step of our hand detection is the selection of the potential hand region. Our simple approach is that it is the minimal region that contains all selected edges. Sometimes these edges can be connected with edges of the periphery. Sometimes unimportant edges near the hand are selected or edges on the body of the user. Therefore, we do not only take the actual found region into account but also the previously found hand regions. To stabilize the region, we compute the median of multiple detected border values of our region. The maximal count of influencing borders is also a parameter.

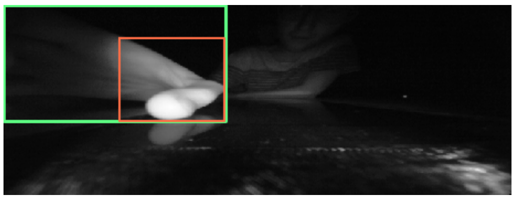
We evaluate our hand regions with the hand labels. We compare the detected region with our label by aid of the function $f(D, L) = 2 \cdot \frac{|D \cap L|}{|D|+|L|} \in [0, 1]$ where F is our detected hand region and L is the hand label. The result we call overlap value. A result of 0 means no overlap of both regions, a perfect match will get a result of 1. In Figure 4.4 examples for different overlap values are given. We assume that our hand labels do match well but are not perfect. Hence, an overlap value larger than 0.95 is a perfect match. An overlap value larger than 0.75 shows large parts of the hand but fewer peripheries. It is possible that parts of the hand are cropped away like the little finger if stretched away in the darker side area. For later finger recognition the most important property of the hand is the touching finger and therefore the lower bound of the detected region.

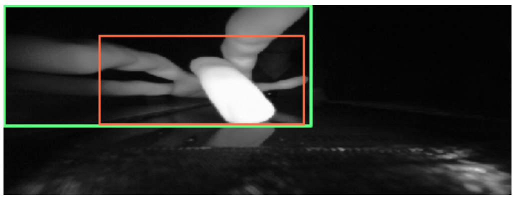
Putting all parameters together we have 22 to train. Three of them are floating point numbers without limits. We are using a train loop to automatically modify and evaluate our configurations. For each parameter configuration we test if an increase or decrease will improve the overall count of detected regions with at least 0.75 overlap value. Thus we divided our database into three parts. We assigned 14 participants as train data, four were used for evaluation. The three left-handers form an extra set because we do not know if their gestures differ from right-handers. Participants who hold the tablet in the hand were split into a train set and test set such that the proportions were as similar as possible.

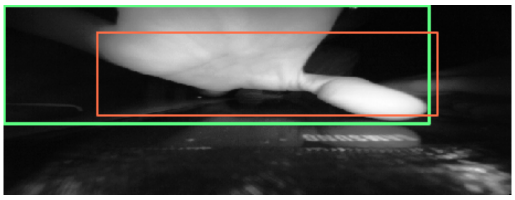
Our train algorithm takes into account how often a parameter was used to search for a better value. It loops over all parameters and tests each with higher and lower values in the valid range. With respect to the context the step width of each parameter is different. At the beginning we chose the steps based on the last digit of our start values. After the training converges we lowered them to a tenth. If this was not possible we chose the next valid value larger than zero. For each direction we tested as long as our evaluation value increased but tested no less than two new configurations. At the end the best of these configurations is taken. The next parameter is chosen by time it was tested. Parameters that were less frequently modified are preferred. With a probability of 12.5% it randomizes the order of the parameters. With another probability of 12.5% the train algorithm will return the best known configuration of all tested possibilities. Thus it will not test the same path of configurations multiple times when the test will not increase the best parameter configuration. This procedure will not guarantee a global maximum but will increase the detection rate on our database as visible in Figure 4.4.

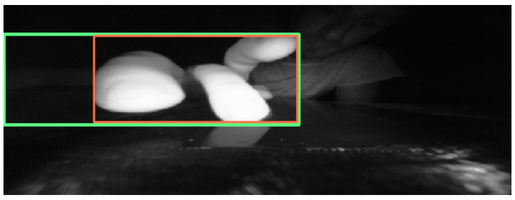

(a) Overlap value 0.3

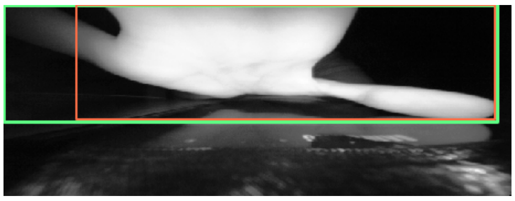

(b) Overlap value 0.4


(c) Overlap value 0.5


(d) Overlap value 0.6


(e) Overlap value 0.7


(f) Overlap value 0.8


(g) Overlap value 0.9

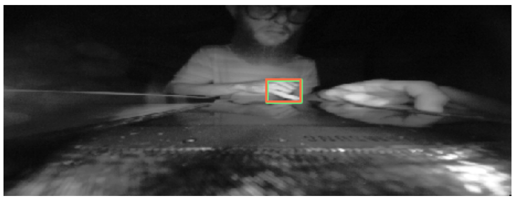

(h) Overlap value 1.0

Figure 4.4: Examples of different overlap values. Manually selected hand regions are green, red regions show the result of a hand detection. We assume that an overlap value higher than 0.75 is enough to recognize the correct finger.

We calculated an overlap value for each image sequence of the first two tasks. In Figure 4.5 our results are shown as box plots, whereby all outliers are shown as red crosses. Interesting points are outliers at the bottom which show that there are sequences where the hand was never found. Also important is the density increase of outliers with increasing overlap value as can be seen at the trained result for the train set and the left-hander set, and especially at all best-of-two results. We see this as a potential for further improvements of the hand detection.

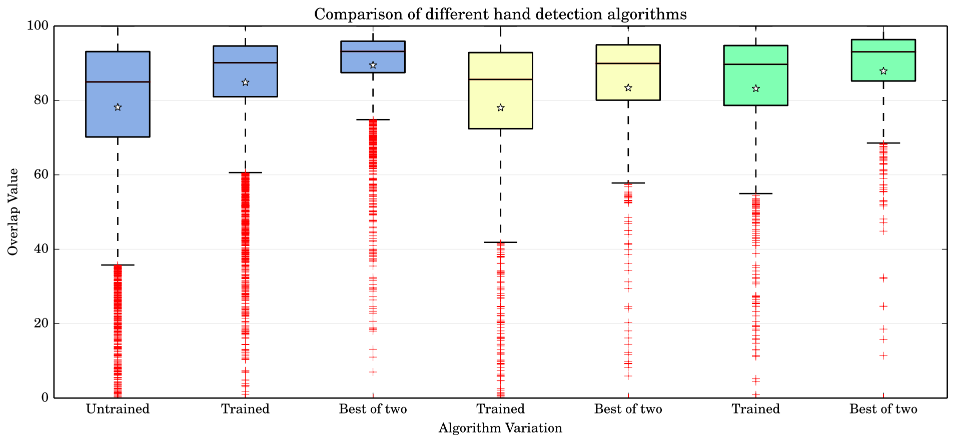


Figure 4.5: Comparison of detection variance between different sets of data. Rightmost the result for the untrained hand detection algorithm on the training set. Then pair wise the results for trained algorithm and best result of two sequences for train data, test data and left hander dataset.

On our first test the algorithm detected 68.43% in the train data set with an overlap value of 0.75 or more. After our training it detects 83.33% of all hands in the train data. In our evaluation set it achieves 71.71% of all hands. Possible reasons for this result are overfitting or our evaluation set selection. Since the LEAP MOTION is a stereo camera we are able to see the same scene from two different positions. Assuming that a bad detected hand will result in a bad finger recognition we tested only taking into account the best of both image sequences in the frame sequence. This increases the results for the same configuration to 92.37% in the train data and to 82.13% in our evaluation set. For the left-handers the detection rate with the trained algorithm achieves 79.81% and 89.01% if only the best of both sequences is taken into account. We see that our algorithm also works for left-handers if they use their right hand.

During analysis of the train results we observed difficult situations that are not handled with our current hand detection method. For example, our algorithm only detects one hand. At this point, a second moving hand results in enlargement of the hand region and therefore in a huge shrink in the overlap value. Other moving and bright objects also confuse the algorithm. We improved this drawback with our position depending filtering which removes objects like bright reflections eyeglasses. Another difficult situation appears if the users move their hand too slowly. In combination with movement of the visible body or background this leads to situations where the optical flow is too ambiguous to select the hand region.

	Thumb	Index Finger	Middle Finger	Ring Finger	Little Finger
Thumb	62.29	19.83	8.35	4.87	4.65
Index Finger	20.27	45.88	21.52	7.73	4.60
Middle Finger	7.42	22.43	47.33	14.40	8.42
Ring Finger	5.57	9.26	19.50	47.66	18.00
Little Finger	6.88	8.06	12.66	23.57	48.84

Table 4.1: Results of first finger recognition approach. The rows stand for the specified finger. The columns show the classification results in percent.

4.2 Finger Recognition

We apply a supervised finger classification algorithm for our finger recognition. To train the classification we use the detected hand regions in our image sequences. Each sequence has a specified finger which we use as a label for the classes.

For our first approach we cropped the hands out of the last images of our sequences where the hand touches the screen. To enable better comparison of the images with different sizes we scale and extend them. The target size is derived from the sizes of the detected results by using the mean. All images are scaled to fit into the target size. We do so because on one hand we do not add too much interpolated data, on other hand we want to keep the loss of details low. It is possible that the aspect ratio is an important feature. We kept it and extended the scaled images with a black border, such that they are centered in an image with our target size. We call this images template images.

To classify a new image, we first scale and extend it like our template images. Then we compare it with each template image by calculating the absolute value of the difference between each pixel. The lower the sum of these differences the higher is the similarity. In the regions where the scaled source images overlap this is true. In all other cases the difference is equivalent to the brightness of the template image, the new image, or black. Using only the region where image and template image overlap results in a loss of information and decreases the correct classified results to a fifth. This is not better than guessing or taking the same finger for all images. To prevent this effect we used the whole image region for detection.

Our database contains about 500 detected hand images for each participant. For each of these participants we used all others to train. Each image was classified and we stored the result in a confusion matrix. At the end we summed up all matrices and normalized them. The result is shown in Table 4.1. The rows stand for the specified finger and the columns for the detected finger. Our aim is to increase the values on the main diagonal.

	Thumb	Index Finger	Middle Finger	Ring Finger	Little Finger
Thumb	73.45	16.11	3.94	3.40	3.10
Index Finger	19.27	66.79	10.20	2.15	1.59
Middle Finger	5.51	13.03	71.24	6.00	4.23
Ring Finger	3.34	2.26	9.43	70.22	14.74
Little Finger	5.12	1.98	4.48	12.57	75.86

Table 4.2: Results of the improved finger recognition approach. The rows stand for the specified finger. The columns show the classification results in percent.

For our second approach we used another data selection and preparation. Instead of adding a black border we extending the scaled image with image periphery. We enlarged the detected region in either x- or y-direction to fit the aspect ratio of the target size. The detected region was kept in the center. However, it is possible that the new region grows beyond the image boundaries. Instead of leaving these regions black we extend the image data. For each pixel where no image data exist we choose the color of the nearest pixel with image data.

We assume that the image contains unnecessary data like background. The LEAP MOTION is a light source and therefore the hand is a bright object in our detected hand region. We selected a threshold that separates background from the hand. For this we used Otsu's method [Ots79] which is implemented in the threshold method of OPENCV. It selects a threshold based on the image histogram and searches for the best separation. All pixel colors higher than the selected threshold where kept. We normalized them to fit the whole possible value range from 0 to 255 by OPENCV's histogram equalization method. The images are rearranged such that the background becomes black and fingers become bright.

The larger values on the main diagonally in Table 4.2 prove that our data preparation works better than using only the cropped hands. We improved the first approach iteratively and tested our new configuration after each change to ensure none will lower the detection rate. However, further increasing the classification rate will result in a better usability. We assume that classifying multiple detected hands in a sequence enables a better classification at touch time by taking the most often detected class. Details are left for future work.

During our finger recognition images of participant 24 received the lowest detection results. We saw that her fingers were not as straight as those of other participants. She also spread out her fingers like only a few other participants and her finger nails were colored. It is possible that there are more hand types or hand posture that are not covered by our participants, for example none of our participants had older hands with wrinkles.

4.3 Summary and Discussion

In this chapter we explained how our finger recognition works. We wanted to use a classifier for finger recognition. Therefore, as a prior step we used hand detection to reduce periphery in our image sequences.

We extracted 280 sequences for each participant from our collected video files. Each sequence contains 20 frames with a left and a right image. The sequences were selected from touch data, such that the last image shows the moment when the participants finger touches the screen. In this frame we manually labeled the hand to enable automatically evaluation of our hand detection.

We treated sequences separately for left and right images. As first step in our pipeline we calculated the optical flow between two images. The edges in the image are determined in parallel. As a second step these edges were filtered by the motion in their image region. Location based filter methods were applied. Thereby we received a set of edges that depend on the hand in consideration. At last stabilization over multiple detected regions in a sequence was applied. Parameters for this hand detection were optimized empirically. Together with the option of using only the best detected hand of both image sequences we received a detection rate of 82.13% in our evaluation.

Our finger detection was trained on the positive results of the hand detection. Thus, we got about 500 of 560 possible images per participant. We use a supervised best fit classification as an approach. As distant function we applied the sum over the difference between two images. Through background removal and stretching of the image histogram we increased the results to over 70% for all fingers but 66.79% for the index finger. We assume that this result is higher in practice if multiple images are classified. This is based on observations because the detected hand region seems to be better before touching the screen.

We also detected that there is an outlier in our finger detection. Participant 24 was classified worse than any other participant. We assume that our hand sample is not large enough for a universal model, especially with respect to the age of the participants.

5 Evaluation

In Chapter 3 we conducted our user study. In Chapter 4 we trained our camera based finger recognition with the collected image data. During the study we also collected data that were not used for training but offered information about finger recognition and its behavior for future users. In this chapter we first analyze measured results and the raw NASA-TLX of our user study. We describe effects we observed and show possible causes. Therefore, we analyzed the measured values of tasks, fingers and our participants' overall workload. Second we discuss the study and possible usage for future application with respect to the feedback. We reported the ideas of our participants, their critics and discuss them.

5.1 Measured Results and Raw NASA-TLX

There is an overall high touch error in our study, as visible in Table 5.1 (a). We assume that our crosses' size of 14×14 mm was too small. As a consequence, they were hidden under the participants' fingers before the contact with the screen. In task 3 and 4 we saw participants tap and hold for a while, because they started without knowing where to draw the line. The starting cross in these tasks was easier to detect because of the hand sign which was larger than the cross itself. Moreover, we observed that especially at drawing tasks the participants focused more on speed than accuracy. For some participants this triggered the appearance of the ANDROID status bar which becomes visible in full screen applications if there is a touch movement from the top side on the screen.

The results of the raw NASA-TLX show a similarity of task 1 and 3 and also a similarity between task 2 and 4. We connected them the following interpretation. To task 1 and 3 we refer as odd tasks and to task 2 and 4 as even tasks. Also we grouped them together in Figure 5.1 and Figure 5.2. In the odd tasks the fingers were used one by one. In even tasks the order of the fingers was randomized.

The order of our tasks was never changed during the study. In Figure 5.1 a clear order in the workload is visible for all aspects asked in the NASA-TLX. We assume that the exceptions of task 4 with respect to temporal demand, effort, and frustration are caused by learning effects. These were possible because we never changed the order of our tasks. During study three participants stated the conditions of this task correctly before we explained it.

	(a) Touch error in mm		(b) Reaction time in ms	
	Mean	**SD**	**Mean**	**SD**
Task 1	11.94	13.50	1106.92	2973.63
Task 2	12.12	13.96	1062.40	214.68
Task 3	37.36	39.38	1173.79	2905.68
Task 4	37.64	39.19	1160.11	270.28

Table 5.1: Measured results of the study. (a) Touch error is the distance between touchpoint and cross, between touch down and start cross, or between touch up and end cross. (b) Reaction time is measured between appearance of the target and next touch down event.

Comparing the tasks with respect to the raw NASA-TLX the largest difference is visible between tapping with one specific finger and random fingers. As participant 13 stated switching fingers is annoying. While task 1 shows a value lower than two in a range from 0 to 21, task 3 is more than three times higher. For line drawing this effect is less explicit. In contrast we observed learning effects at temporal demand, effort, and frustration. Possibly these learning effect are the reason why the distance between the drawing tasks is smaller than for tapping tasks. These results, however, show that switching fingers often decreases the comfort. The results of performance also show that our participants felt more insecure about their results. Our collected data in Table 5.1 (b) supports this. We assume that the little improvement in the measurement of task 4 is caused by the tasks conditions. In task 3 our participants remembered the specified finger at the beginning. In task 4 they need to identify the start cross before they knew the stated finger. As a result they took more care in task 4 and less confusion errors appeared.

The temporal demand shows the fewest change over all tasks. We did not tell our participants that there was no time limit set but we asked them to carry out the task as early as possible. The participants were able to do the study in a very high speed or to take as much time as needed. The small difference between the tasks shows that tapping with changing fingers instead of only one seems to be slower. Our measured results, however, showed only a difference of a few milliseconds. We did not take the first cross into account because at the beginning the task was started during the explanation of the study. The variation for each task seems to be dominated by the participant's different speed. For example, participant 5 who stated very good touch screen experience was twice as fast as participant 12 who was left-hander and used the right hand. In the randomized tasks the faster participants where more inhibited than the slower which results in a lower variation. They provide us with an interesting view on our tasks. Especially the odd tasks are equal in all aspects of the NASA-TLX but frustration. A similarity can also be found between the even tasks. We never changed the order of the tasks

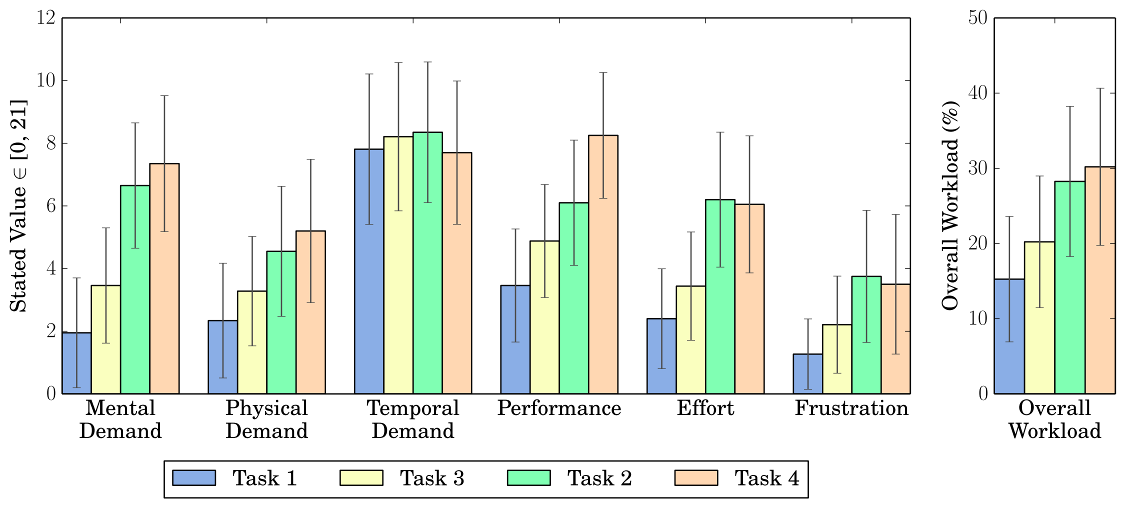


Figure 5.1: Results of the twelve raw NASA-TLX for each task presented with standard error. Bars for task 1 and task 3 present the mean of all five NASA-TLX in these tasks. The y-axes do not show the whole possible value range.

and therefore learning effects are a possible reason. The same effect can be seen at temporal demand and effort but only for the even tasks where the specified finger was randomly selected for each new target. The overall workload also shows that both tasks are twice the overall workload of the other tasks. In the even tasks the mental demand is over three times higher than in the odd tasks. We summarize that switching fingers needs much more time than using only one.

The overall value for each of the four tasks in our study lays in the lower third of the possible range. For the odd tasks where only one finger was used per subtask we assume that the result represents the minimum tapping effort for our study tasks. The difference between odd and even tasks results from switching fingers and all of its side effects.

To measure the overall workload for each finger we only took the odd tasks into account. After each subtask we asked our participants to fill out a NASA-TLX. Each could be mapped to exact one finger. For the even tasks this is not possible because the NASA-TLX contains information about all fingers. We show the results in Figure 5.2. Interestingly the ring finger's overall workload is lower than the overall workload of the index finger for tapping. We see the large difference between tapping and line drawing for the thumb as important for user interface design. The thumb seems to be good for tapping on tablet sized screens but not for line drawing tasks. This may be different for global drawing tasks like scrolling. Also interesting is the small difference for the little finger. Especially the low value for line drawing since five of our participants stated in the feedback that this was one of worst finger for tapping. The ring finger was also stated as a bad finger for touching by four of our participants. The thumb got two bad opinions while the index finger and the middle finger were only mentioned by

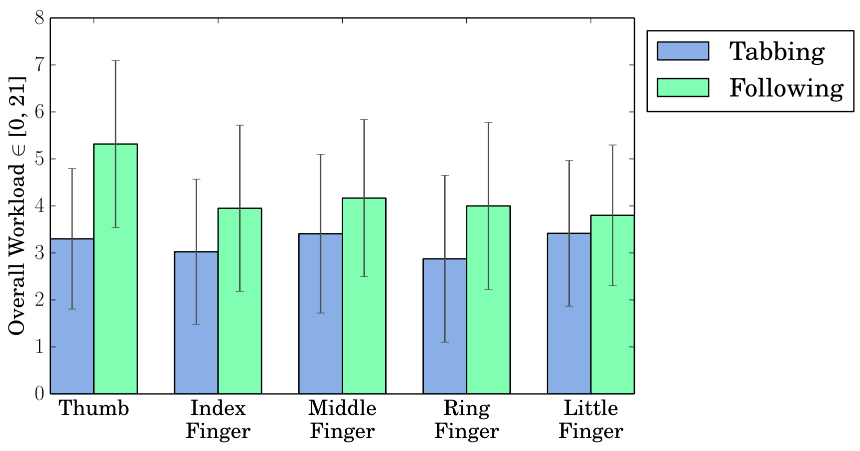


Figure 5.2: Overall workload with standard error for finger for tapping and line drawing with a specific finger.

one participant as good fingers for touching. Our oldest participant stated that some fingers are good for tapping while others are better for drawing. Figure 5.2 does not support this statement but the measured results show what he meant. In his case the thumb was slightly better for drawing than for tapping. Ring finger and little finger were clearly better in tapping. During the study we observed that participants lost the contact with the touchscreen more often while using this fingers.

5.2 Feedback

The feedback about specific fingers was handled in Section 5.1. In this section we analyze all other written and oral feedback of our participants. The feedback questionnaire handles feedback about the study itself and our participants' ideas about where finger recognition is supporting users.

Our participants had various ideas of how finger recognition can be used as input. Some did not clarify what they meant and it is possible that they only filled in some text. Sociological reasons like embarrassment are also possible. For example, one participant stated that finger recognition can be used for "better interface design". For the study itself this phenomenon also appears. Instead of leaving an empty field two participants wrote that "[the study] was good". However, one participant stated that he would not use it because he does not like touchscreens in general. Also he stated that typing on touchscreens is more error-prone, like proved by Sears et al. [Sea+93]. He also did not understand why we used a NASA-TLX. He did not see any physical demand. The NASA-TLX was also mentioned by another participant who criticized the option count per question. He suggested to use only five or seven steps or a continuous slider. Thus he wanted to ease the task for participants. For two other participants the study seemed to be boring. They stated that background music or making the study more game-like would improve the joy of the participants. We do not know which effects this would have on the participants. It is possible that results increase but games do often work with stress and therefore may increase errors and frustration.

Some ideas were given by multiple participants and therefore we see them as possible test cases in future work. Six participants stated that games could profit by finger recognition. Thus, less space for controls is needed and more is left for content presentation. There is another possible reason that tapping on the correct onscreen button takes more time or is more error prone than hitting a button on a hardware keyboard. Specific finger input may be as fast as hitting a hardware button but this requests the detection of multiple fingers.

Another stated possibility is a finger depending context menu as stated by two of our participants. Not all fingers need to open another menu one person stated, thus it is usable like a mouse. Also gesture detection was mentioned. On the one hand finger depending gestures certainly need more cognitive load, on the other hand the same movement can be assigned to a finger dependent function. The last option reduces required space and increases accuracy for gestures. Thus more locally dependent gestures are possible. One person stated that finger recognition is usable as quick start for programs. We think that this idea can be extended to any global function like opening an app starter similar to the Windows start menu. This menu opens if the Windows key is pressed without other keys.

Participant 4 stated that he would use finger recognition to improve onscreen keyboards with a finger depending correction. Another stated that finger recognition may be usable for authentication. Thus users are able to authenticate themselves without seeing the screen. It is possible that such an authentication can be hidden more easily from observers. The ideas of our participant 14 is probably based on the idea of using a touchscreen device without explicitly looking at the screen. He would use it for remote control software in a railway set, industry or music player. Map navigation can also profit from finger specific actions.

Nevertheless, our finger detection is probably not acceptable by left-handers. Two of our three left-handers mentioned that the usage of the right hand affected the study in a negative sense. We additionally assume that enabling the detection of multiple hands together with separation of left and right hands increases significant the usability.

Two participants stated that the camera is too big and disturbs. We know that the LEAP MOTION sensor is too big for an all-day use of our prototype. Possible improvements are integration of a camera in the device border or utilizing hinges are an option. We showed that using the best of both camera images increases our hand detection results. If multiple cameras are integrated in the devices border, for example into the edges, the operation of the system may be simplified. Multiple cameras which cover the area over the screen may also solve the problem of hidden fingers.

5.3 Summary and Discussion

In this Chapter we analyzed the results of the study that are not used for our finger recognition. We interpreted the results of the raw NASA-TLX, measured results and feedback of our participants.

Our measured results show a high targeting error, compared to the error measured by Holz and Baudisch [HB11]. Possible reason is the size of the crosses which is 14mm $\times$ 14mm and therefore hidden at the touch moment. We also observed that especially at drawing tasks the participants focus laid more on speed than accuracy.

The results of the raw NASA-TLX showed a similarity between tasks with randomized finger and those without. Therefore, we grouped them as odd tasks which were task 1 and 3 where no randomization took place. The other tasks form the even group where finger randomization was used.

We observed that our tasks are much easier to handle without randomization as shown in Figure 5.1. Our randomization changes the stated finger we asked the participants to use and increases especially mental demand and frustration. Also we observed learning effects in our results. This is possible because the order of our tasks was never changed during the study. In the first two tasks our participants learned that a randomized task follows the ordered task. In the third they learned that it was the same like the first one but with line drawing instead of tapping. Thus they assumed conditions of the fourth task correctly. The learning effect is visible at temporal demand, effort and frustration.

Our per finger view on the raw NASA-TLX in Figure 5.2 shows that tapping is easier than line drawing. This effect is largest for the thumb and smallest for the little finger. Moreover, based on statements of our participants, we achieved an order of fingers. From best to worst this order is index finger, middle finger, thumb, ring finger, and little finger, whereby the biggest gap is between thumb and ring finger. The latter takes twice as much bad statements.

In the feedback two applications for finger recognition were often stated. First was interaction with a game. It is possible that our participants do not want to have controls in games which shrink the content area. Also a finger specific action is possibly less error-prone and therefore less frustrating. Second idea were finger specific menus in programs which also reduces control space. Other finger specific functions, like a global program start, where mentions but often not clarified.

6 Conclusion and Future Work

In this chapter we summarize the content of this work. We describe what we did and our results of finger recognition and participants behavior. Our ideas about future work with finger recognition are presented in the second section.

6.1 Conclusion

Nowadays interaction with touchscreens is restricted. Our goal is to extend the input dimension by taking the finger type into account. Colley and Häkkilä [CH14] tested users' behavior with such methods. While their used prototype was static we want to test if finger recognition is possible on mobile devices.

We designed a prototype by combining a Leap Motion sensor[1] (LEAP MOTION) with a tablet device. The LEAP MOTIONs manufacturer already implemented hand detection methods, however, these methods do not apply to our prototype. We described our mobile prototype and how we detect hands and fingers. We mounted a LEAP MOTION at the top of a tablet. We conducted a study where we used this prototype, a laptop device that recorded videos from the LEAP MOTION, and a desktop PC which shows NASA-TLX and other questionnaires like feedback. The goal of the study was the recording of data to train hand detection and finger recognition for our given conditions.

Our study was structured into four tasks. We asked our participants to tap at cross targets. For each target a finger was indicated which was used as label in our finger recognition. The first task was separated into five subtasks whereby each subtask has its unique indicated finger for forty targets. In the second task eighty targets were shown one by one. The indicated finger changes randomly from target to target until each finger was used 16 times. Task three and four were similar to the first both but tapping was replaced by line-drawing. Therefore we added a second cross to mark the end of the line.

We received 280 tapping scenes for each participant. From recorded video files we extracted sequences of 20 frames whereby the last frame shows the touch down event of a touch scene. Each of the frames contains two images with a resolution of 640×240 pixel. The LEAP MOTION

[1] http://leapmotion.com

offers only the infrared channel of taken images. Hence, skin color cannot be used to detect hands as the LEAP MOTION has an integrated infrared light source. Hands appear as bright objects in the field of view. We apply an edge detector to each image and approximated the optical flow in parallel. Visible edges were filtered by motion and position. The hand region was derived from the remaining edges. At last the detected regions of multiple frames were stabilized by calculating the median. Parameters for this method were empirically improved. Our method achieves 71.71% of all hands in the evaluation set reasonable good for finger recognition.

Our finger recognition is based on the results of our hand detection. We use detected hands as template images. To increase our detection results we prepared each image by extending the region to fit the aspect ratio of a target size. As target size we use the mean of all detected hand sizes. All template images are scaled to the target size and normalized by applying a threshold and histogram equalization. To classify a new image, we first applied the same operations. As second step we determine the template image with the smallest distance whose label gives us information about the touching finger. This classification is based on the assumption that hands with the same touching finger are similar to each other. Our results also show a detection rate of over 70% for all fingers but the index finger.

In the evaluation of feedback and measurements in the study we found that using another finger for each target increases the mental demand and frustration by a factor larger three. However, we also observed learning effects which decreased the factor especially visible for temporal demand, effort, and frustration. Also in the second half of our study the differences were smaller. Participants stated that not all fingers are comfortable for touching. Best ratings were given for index finger and middle finger, worst were given for the little finger. User interfaces may use this behavior e.g. to assign critical functions to uncomfortable fingers like deleting with the little finger.

All in all, the feedback of our participants in the study showed that there is an interest in fingers as input dimension. Most of our participants stated that it would be useful for open different menus or as input in a game. Games often use the whole display area to show their content. Displayed controls will crop space which can be freed if fingers replace the function.

6.2 Future Work

Based on observations during the evaluation we found that the best moment to detect the hand appears before the actual contact with the screen. As a result, we took sequences before the touch event. If this assumption is true or does only hold for special cases needs to be tested. We assume that there is a time range between the decision of the user to take a specific finger and the touch event of this finger. While our work focuses on the finger recognition possibility we assume that refining the time range for a detection may improve the results.

As an alternative for our hand image classification we tested multiple machine learning algorithms. Best results were achieved by a neuronal network with two hidden layers. This network had a high error rate of 50% and did not improve during training. We assume that the distortion makes it hard to learn which features are relevant. Neuronal networks are learning from data. Since we trained with only 560 touch scenes from each participant it is possible that the remaining data set is too small. A larger data acquisition study may help to improve the learning.

The question how finger specific input should be used is discussed in this work and by Colley and Häkkilä [CH14]. However, it is studied with a small number of participants under lab conditions. Concentrating on the fundamental challenges works better this way. In contrast, everyday life has a far higher range of challenges. Changing conditions in background structure, environment light and motion of different objects in the field of view need to be handled. Building a database with a wide range of labeled image data may help to refine our method. Variables can be participants age, gender, skin brightness and if they use the left or right hand. Furthermore, data should be collected with and without background motion, in dark and bright environments, and with multiple fingers at once.

Two participants stated that the LEAP MOTION is to big for an all day usage. Integration into the device is necessary for comfort of future users. We think about multiple possible solutions. If devices, like smartphones, have a curved touchscreen, it may be possible to integrate cameras near the screen corners. This way they are able to observe the screen without sticking out of the device. If the camera does not have to watch all the time, hinges may be used to hide it inside the device casing. It is also thinkable that no extra camera is needed. Today's mobile devices which contain a front camera may be modified with a small mirror or a lens to enable an observation of the touching fingers.

We mentioned that our method may be used in smartphones. Today the range of devices which contain a touchscreen is big. There are laptops, ticket machines, tablets, smartphones, smartwatches and more. We assume that especially devices with small screens may profit from finger recognition as additional input. On laptops and tablets it may be used like a mouse, on smartwatches it can be reasonable to assign program wide actions to fingers. User interfaces may change through finger recognition, but scalability is problematic. The user needs to know what kind of action hides behind which finger. Most mouses and touchpads offer left click, right click and scrolling. Users learned that left clicks invoke an action at the mouse position and that the right button opens a context menu. Hence, we assume that learning two or three finger dependent functions is not a hard challenge for experienced touchscreen users. User-centered design studies can test learnability and limits of this technique.

Bibliography

[Aze+12] S. Azenkot, J. O. Wobbrock, S. Prasain, and R. E. Ladner. “Input Finger Detection for Nonvisual Touch Screen Text Entry in Perkinput.” In: *Proceedings of Graphics Interface 2012*. GI ’12. Toronto, Ontario, Canada: Canadian Information Processing Society, 2012, pp. 121–129. ISBN: 978-1-4503-1420-6. URL: http://dl.acm.org/citation.cfm?id=2305276.2305297 (cit. on p. 17).

[BA15] D. Buschek and F. Alt. “TouchML: A Machine Learning Toolkit for Modelling Spatial Touch Targeting Behaviour.” In: *Proceedings of the 20th International Conference on Intelligent User Interfaces*. IUI ’15. ACM. 2015, pp. 110–114. DOI: 10.1145/2678025.2701381 (cit. on pp. 15, 19).

[BRMS13] D. Buschek, S. Rogers, and R. Murray-Smith. “User-specific Touch Models in a Cross-device Context.” In: *Proceedings of the 15th International Conference on Human-computer Interaction with Mobile Devices and Services*. MobileHCI ’13. ACM, 2013, pp. 382–391. DOI: 10.1145/2493190.2493206 (cit. on pp. 15, 19).

[Bra00] G. Bradski. “The OpenCV Library.” In: *Dr. Dobb’s Journal of Software Tools* (2000). URL: http://www.drdobbs.com/open-source/the-opencv-library/184404319 (cit. on p. 23).

[CH14] A. Colley and J. Häkkilä. “Exploring Finger Specific Touch Screen Interaction for Mobile Phone User Interfaces.” In: *Proceedings of the 26th Australian Computer-Human Interaction Conference on Designing Futures: the Future of Design*. OzCHI ’14. ACM. 2014, pp. 539–548. DOI: 10.1145/2686612.2686699 (cit. on pp. 12, 15–17, 19, 21, 45, 47).

[CT98] R. Cutler and M. Turk. “View-Based Interpretation of Real-Time Optical Flow for Gesture Recognition.” In: *Third IEEE International Conference on Automatic Face and Gesture Recognition*. Vol. 0. IEEE Computer Society. 1998, p. 416. DOI: 10.1109/AFGR.1998.670984 (cit. on pp. 14, 20, 29).

[Can86] J. Canny. “A computational approach to edge detection.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8.6 (1986), pp. 679–698. DOI: 10.1109/TPAMI.1986.4767851 (cit. on pp. 18, 31).

[Che+11] Y.-L. Chen, W.-Y. Liang, C.-Y. Chiang, T.-J. Hsieh, D.-C. Lee, S.-M. Yuan, and Y.-L. Chang. “Vision-based Finger Detection, Tracking, and Event Identification Techniques for Multi-Touch Sensing and Display Systems.” In: *Sensors* 11.7 (2011), pp. 6868–6892. DOI: 10.3390/s110706868 (cit. on p. 16).

[Far03] G. Farnebäck. “Two-Frame Motion Estimation Based on Polynomial Expansion.” In: *Image Analysis*. Ed. by J. Bigun and T. Gustavsson. Vol. 2749. Lecture Notes in Computer Science. Springer, 2003, pp. 363–370. DOI: 10.1007/3-540-45103-X_50 (cit. on p. 30).

[HB11] C. Holz and P. Baudisch. “Understanding Touch.” In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’11. ACM. 2011, pp. 2501–2510. DOI: 10.1145/1978942.1979308 (cit. on pp. 14, 19, 44).

[Har06] S. G. Hart. “NASA-Task Load Index (NASA-TLX); 20 Years Later.” In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. Vol. 50. 9. Sage Publications. 2006, pp. 904–908. DOI: 10.1177/154193120605000909 (cit. on p. 24).

[IB96] M. Isard and A. Blake. “Contour tracking by stochastic propagation of conditional density.” English. In: *Computer Vision — ECCV ’96*. Ed. by B. Buxton and R. Cipolla. Vol. 1064. Lecture Notes in Computer Science. Springer, 1996, pp. 343–356. ISBN: 978-3-540-61122-6. DOI: 10.1007/BFb0015549 (cit. on pp. 18, 20).

[KNR08] S. K. Kang, M. Y. Nam, and P. K. Rhee. “Color based hand and finger detection technology for user interaction.” In: *International Conference on Convergence and Hybrid Information Technology*. ICHIT’08. IEEE. 2008, pp. 229–236. DOI: 10.1109/ICHIT.2008.292 (cit. on pp. 17, 18, 20).

[KT04a] M. Kölsch and M. Turk. “Fast 2D Hand Tracking with Flocks of Features and Multi-Cue Integration.” In: *Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop*. Vol. 1. CVPRW’04. IEEE. 2004, pp. 158–158. DOI: 10.1109/CVPR.2004.71 (cit. on pp. 17, 20).

[KT04b] M. Kölsch and M. Turk. “Robust Hand Detection.” In: *Proceedings of the Sixth IEEE international conference on Automatic face and gesture recognition*. FGR’ 04. IEEE. 2004, pp. 614–619. DOI: 10.1109/AFGR.2004.1301601 (cit. on pp. 18, 20).

[LL11] D. Lee and S. Lee. “Vision-based finger action recognition by angle detection and contour analysis.” In: *ETRI Journal* 33.3 (2011), pp. 415–422. DOI: 10.4218/etrij.11.0110.0313 (cit. on pp. 19, 20).

[ML14] W. McGrath and Y. Li. “Detecting Tapping Motion on the Side of Mobile Devices by Probabilistically Combining Hand Postures.” In: *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*. UIST ’14. ACM. 2014, pp. 215–219. DOI: 10.1145/2642918.2647363 (cit. on p. 14).

[MM04] J Mackie and B McCane. “Finger detection with decision trees.” In: *University of Otago, Department of Computer Science* (2004), pp. 399–403. URL: http://www.cs.otago.ac.nz/staffpriv/mccane/publications/ivcnz04-mackie-1024.pdf (cit. on pp. 19, 20).

[Mar+11] N. Marquardt, J. Kiemer, D. Ledo, S. Boring, and S. Greenberg. “Designing User-, Hand-, and Handpart-aware Tabletop Interactions with the TouchID Toolkit.” In: *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*. ITS ’11. ACM. 2011, pp. 21–30. DOI: 10.1145/2076354.2076358 (cit. on pp. 17, 19).

[Ots79] N. Otsu. “A Threshold Selection Method from Gray-Level Histograms.” In: *IEEE Transactions on Systems, Man and Cybernetics* 9.1 (1979), pp. 62–66. ISSN: 0018-9472. DOI: 10.1109/TSMC.1979.4310076 (cit. on p. 37).

[RYZ11] Z. Ren, J. Yuan, and Z. Zhang. “Robust hand gesture recognition based on finger-earth mover’s distance with a commodity depth camera.” In: *Proceedings of the 19th ACM international conference on Multimedia*. MM ’11. ACM. 2011, pp. 1093–1096. DOI: 10.1145/2072298.2071946 (cit. on pp. 18, 20).

[Rav+09] J Ravikiran, K. Mahesh, S. Mahishi, R Dheeraj, S Sudheender, and N. V. Pujari. “Finger detection for sign language recognition.” In: *Proceedings of the International MultiConference of Engineers and Computer Scientists*. Vol. 1. IMECS 2009. 2009, pp. 18–20. URL: http://www.iaeng.org/publication/IMECS2009/IMECS2009_pp489-493.pdf (cit. on pp. 18, 20, 29).

[Rog+11] S. Rogers, J. Williamson, C. Stewart, and R. Murray-Smith. “AnglePose: Robust, Precise Capacitive Touch Tracking via 3D Orientation Estimation.” In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’11. ACM. 2011, pp. 2575–2584. DOI: 10.1145/1978942.1979318 (cit. on pp. 15, 19).

[Sea+93] A. Sears, D. Revis, J. Swatski, R. Crittenden, and B. Shneiderman. “Investigating touchscreen typing: the effect of keyboard size on typing speed.” In: *Behaviour & Information Technology* 12.1 (1993), pp. 17–22. DOI: 10.1080/01449299308924362 (cit. on p. 42).

[Yan+13] X.-D. Yang, K. Hasan, N. Bruce, and P. Irani. “Surround-see: Enabling Peripheral Vision on Smartphones During Active Use.” In: *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*. UIST ’13. ACM. 2013, pp. 291–300. DOI: 10.1145/2501988.2502049 (cit. on p. 14).

All links were last followed on November 3, 2015.

Declaration