Universität Potsdam
Hasso-Plattner-Institut für Softwaresystemtechnik

Diploma Thesis

Thumb on Hand Interaction

by

Christian Loclair

Prof. Dr. Patrick Baudisch

Christian Holz

Potsdam, July 2011

DISCLAIMER

I certify that the material contained in this thesis is my own work and does not contain significant portions of unreferenced or unacknowledged material. I also warrant that the above statement applies to the implementation of the project and all associated documentation.

Christian Loclair, Student
Potsdam, Germany
July 2011

1

ACKNOWLEDGEMENTS

I am greatly thankful to Prof. Dr. Patrick Baudisch for the possibility to study HCI. His scientific support, trust and overwhelming enthusiasm during the course of this thesis, continuously pushed me to do my best. His qualities created an incomparable learning environment. I would like to acknowledge the invaluable guidance and encouragement received from my advisor, Christian Holz. His visions, encouraging milestones and motivation in this work has been great to me and completely changed my perspective on how to attend and accomplish tasks. I am also highly thankful to Sean Gustafson for his support, for sharing thoughts and his scientific experience. He did a tremendous and irreplaceable work for the *Pinchwatch* Paper which represented my foundation for further work and finally motivated this thesis. Being his colleague and sharing the office was inspiring, interesting and fun at all times. I would also like to thank all of the members of the HCI lab. Especially Anne Roudaut, Henning Pohl, Frederik Rudeck, Torsten Becker and Daniel Bierwirth has been great companions. It was a great honor to have the possibility to learn from them. I want to thank my friends for always having an open eye for me and sharing thoughts. Especially the project group Animatronik, Niels Robitzky, Christian Lützow and Rosalina Stuckert had a tremendous influence. Last but not least i want to thank my parents, my family and Constantin Hieronimi. Their unbreakable trust and support at all times made everything possible and represent my backbone.

2

ABSTRACT

Interaction technologies for mobile devices became an important research sector, over the past years. A great number of inspiring work concerning different concepts and investigations about the humans abilities to interact has been done. However commercial mobile interfaces are still based upon input technologies, which have been established for desktop scenarios. Pointing on touchscreens and performing input on keyboards demand the user's attention, yet only represent secondary tasks in mobile scenarios. People being on the go desire to fulfil a primary task, such as getting to the office or relaxing in the park. Hence the Distraction caused by mobile devices is unneeded and sometimes even dangerous. This thesis proposes thumb on hand gestures for interacting with mobile devices. Our concept address the following contradictory need. The user's hand should be exempted from a physical device. At the same time we want to enable the user to have his device "at hand" at all times. We present an optical detection system and the relevant mechanisms to identify and analyse the user's gesturing hand. Therefore our solution facilitate the possibility to quickly interact with imaginary input widgets in order to control mobile devices. Furthermore we present different types of applications which are controlled by single hand gestures and demonstrate the efficiency of our implemented solution.

CONTENTS

3

INTRODUCTION

3.1 INTRODUCTION

An increasing amount of human computer interaction research concentrates on various disciplines for mobile scenarios. This trend is based on rapid and fundamental changes in this sector. Today's Smart-phones merge the functionality of a hand-held computer, like managing calendars or browsing the internet into the classical functions of a mobile telephone. This coalition requests an innovative interaction-design that cant be adopted from stationary devices and must be suitable for mobile scenarios. Low priority tasks like checking mails on our mobile device shouldn't distract us from our primary tasks like crossing the streets or waiting for the bus. In order to minimize this distraction, the interface needs to reduce the amount of attention mobile devices command. A satisfying interface-concept, which is suitable for mobile scenarios and actualizes the required reduction, has not yet been implemented.

This thesis proposes thumb and hand interaction as input for mobile devices. We present an optical solution illustrated in Figure 1, detecting single hand gestures. Our main focus is to implement rich input options generated from the thumb interplaying with the corresponding hand. We suggest gestures like pinching fingers or moving the thumb across the palm as appropriate for quick interactions in mobile scenarios and present the corresponding realization techniques. Our design involves a chest mounted camera system, enabling the detection and analyse of the users hand as it is held in front of the body. The tracking of the thumb's position and motion in reference to the users empty hand enables the possibility to control imaginary widgets like sliders,dials or buttons placed along our Hand.

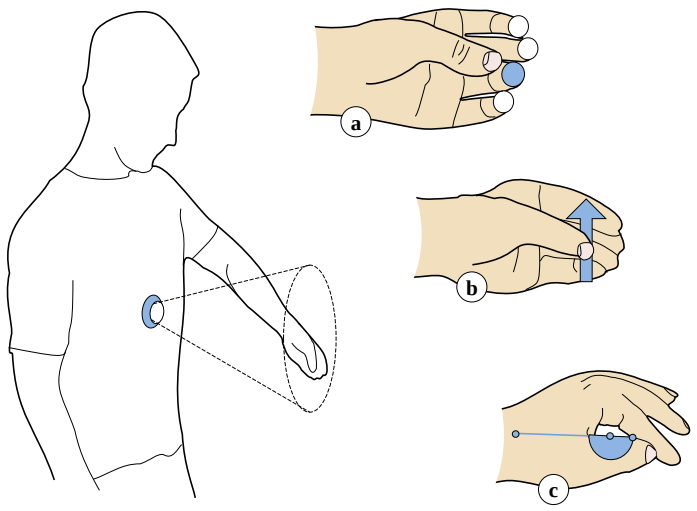

Figure 1: Thumb on hand interaction: The user interacts with the device by performing single hand gestures in front of his body. We implemented interactions like pinching individual fingers (a), using the tip of the thumb in order to slide across the hand (b) or using the index finger in order to rub along the thumb (c). A chest mounted camera enables the possibility to detect the desired gestures.

Our design directly address the individual factors, which determine the amount of attention that mobile devices command. The amount of attention a device commands is aggregated by the time we need to access the device,its usage time and the intensity of our attention we have to pay the device. Cui et al. published a study,finding that 30 percent of men and 40 percent of woman miss calls on their mobile phone simply because of the fact that they cant access their phone fast enough. Having our input-space at hand at all time minimizes the access time to our mobile device. The usage time typically consists of a user navigating through a menu structure in order to activate the desired procedure. Rich diversity of functionality offered by temporary devices, restricts interaction minimization. However the reduction of the timeslot for frequently appearing operations is appropriate. Single purpose Gestures like pinching index finger for switching to the next song in our play-list or simply sliding across the palm for changing volume minimizes the usage time of our device in many common cases. Smart phones, as they are typically

equipped with a touch screen, don't offer the possibility to interact eyes free, since they provide no tactile feedback. This condition increases the intensity of attention the device commands, because it bends the users eye to the touch screen. The hand is very sensitive to tactile feedback, as we are used to operate with it in order to feel out, touch and hold objects. This feature can be used to create an eyes free interaction and therefore reduce the intensity of attention. Dan Ashbrooked defined Micro Interactions, interactions that are no longer then 4 seconds, as appropriate for interacting with mobile devices. By decreasing all of the factors that determine the amount of attention a device request, those quick interaction-bursts can be enabled for mobile scenarios. Our Design realizes the necessary features, offering micro interaction technologies.

Walktrhoug

In order to illustrate the efficiency of our input concept we introduce a system consisting of a chest mounted depth camera in order to detect gestural input. The input commands control a small mobile device, which is equipped with a bluetooth mini headset for acoustic feedback and communication. Our system is illustrated in Figure 2 (a) and could become advantageous in the following scenario.

Joe is rock climbing with friends and wears his device, which runs a climbing application. Right in the first sequence of the climb, he decides to hear classical music in order to calm down. Hence he moves his hand in front of his chest and spreads his fingers (b). This gesture activates the input detection of the system and enables Joe to perform a pinching gesture (c) to evoke the playback. Joe wants to pay attention to his friends commandos and his environment, therefore he decides to turn down the volume a little bit. He simply does so by sliding his index finger softly along the thumb (d). Since our system offers single hand gestural control, Joe is able to claw at the wall with the other hand the hole time. The device turn to its passive state again, as Joe's hand leaves the field of view of the camera. Having passed the first intense climbing passage, Joe

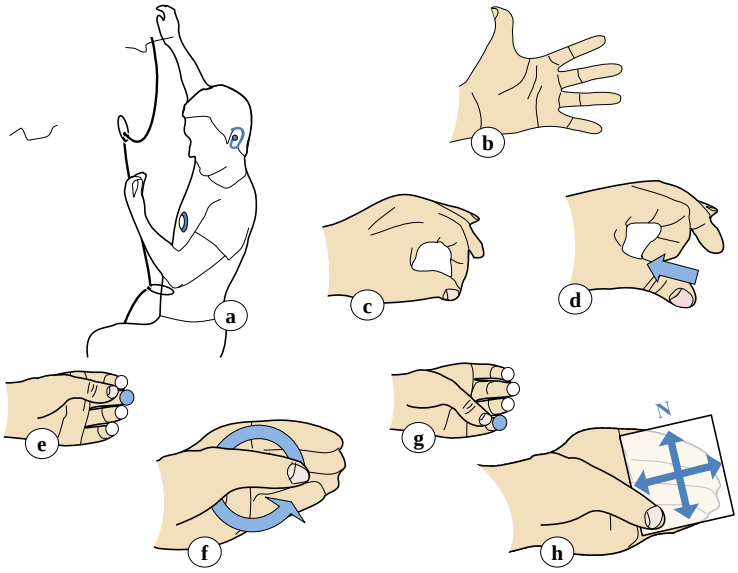

Figure 2

decides to rate the technical difficulty of his last exercise by the use of his climbing application. He activates the device quickly and pinches the middle finger (e), representing the rating option of the application. An acoustic signal indicate the successful access to the desired functions. Joe continues by pinching his thumb on to the hand and imagines a dial to be placed across his hand. He is able to rate the difficulty of his last maneuver by turning this dial in the corresponding direction (f). Joe performs this interaction eyes free, still keeping an eye on the wall and his environment. The tactile abilities of his hand provide all the necessary feedback for Joe to register the current state of his input. Joe has finished his rating input and wants to continue to climb. This time he is not able to move the hand out of the field of view of the camera since the next grasp of the wall is directly in front of his chest. In order to turn off the device without moving his hand out of the frame, Joe quickly spreads his fingers again. This action turns the gesture recording off and avoids further unwanted input. Later up on his route Joe registers a thunder far away. He is well aware of the fact that weather conditions are mission-critical informations for climbers. His lack of ability to locate the active front from his current position motivates him to use the weather function of his climbing application. Therefore he simply

perform the activation gesture, taps the little finger representing the map function and navigates across his hand. He imagines his hand to represent a map with the corresponding cardinal directions. As he moves his thumb tip across the finger's surface he virtually senses different directions. An increasing acoustic frequency indicate Joe to be close to the point which represent the direction the active front is located. All of a sudden his foot looses stance and slips a little across a wet stone. Joe is able to immediately hold close to the next edge and get a secure hold, since his hands were exempted from a physical device at all times. Finally he continuous to study the map, finds the bad weather conditions to be located south and decides to climb down to his base-camp.

This scenario presents multiple benefits offered by our input technology.

- Using a TOF depth camera is well founded for mobile scenarios, in its ability to separate the relevant data from the irrelevant background. Temporary Time-Of-Flight camera's provide robust information of the captured area even under extreme conditions of natural light flooding the observed stage. Hence a changing environment don't necessarily effect the quality of the desired data in a significant manner.

- Unlike a physical device the user has to hold in his hands at all times in order to perform input, our solution offer the possibility to interact with imaginary widgets. The hands remain empty at all time, enabling the user to immediately return to his primary task.

- All of the presented interaction types are based upon the thumb being in contact with certain joints of the hand. Because of that, the user can make use of the tactile features of his hand, giving precise feedback about the desired process of input. Therefore our solution enables complete eyes free use. In the presented scenario this is not only more comfortable, but even more secure then interacting with a devices which demand the user's visual attention.

- We presented multiple imaginary widgets, offering versatile input options. Hence the user is able to perform all of the presented single purpose gestures by single hand gestural input. The other hand is allowed to interact with the physical world at all times.

3.2 CONTRIBUTIONS

This thesis presents a mobile gesture interaction system. Our implementation enables different single hand gestures to be used for different types of applications. We present two major contributions, which occupy two different layers within our system's architecture. The cooperation of those layers is illustrated in Figure 3.

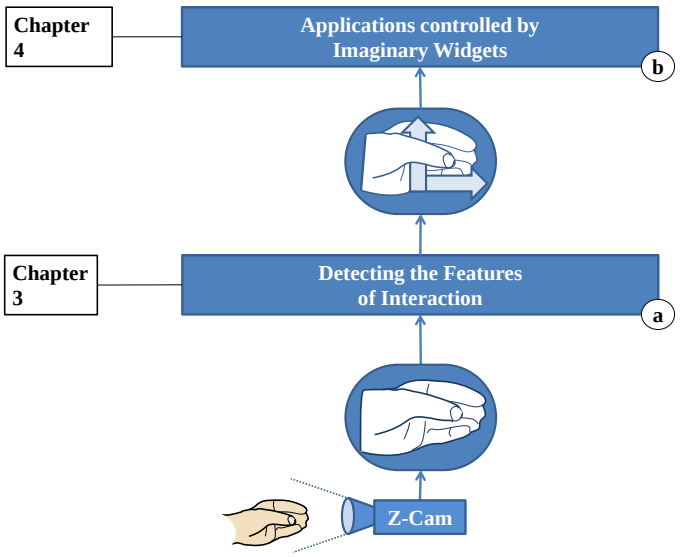

Figure 3: Overview of system's architecture representing our contributions

- (a) We implemented a feature detection layer analysing the characteristics of the user's gesturing hand. Image data, captured by the depth camera, is analysed by this process in real time. Our Algorithm detects detailed features of the hand, the posture and calculates the thumb's relative position in reference to the hand and fingers.

- (b) We implemented different imaginary Widgets for single hand gesture interaction. The resulting data of the feature detection layer is translated into those widgets, enabling access to continuous and discrete input of the corresponding applications.

The following subsections contain a more detailed description of our contributions.

3.2.1 *Detecting the Features of Thumb and Hand Interaction*

Our implementation extracts detailed characteristics of a gesturing hand. The following Figure 4 illustrate a sample of our feature detection, represented by debug output images.

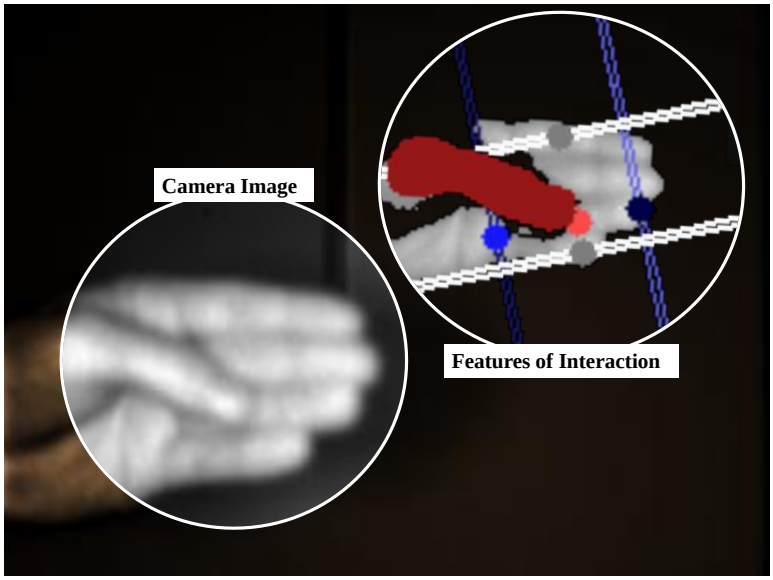

Figure 4: The Thumb operating on the hand: The main frame displays the camera's image of the user's hand. The debug image highlights the user's thumb within a reference system, which is fixed to the hand.

A chest mounted Time of Flight Camera observing single hand gestures, provides the essential data for the reconstruction of the performed gestures. We present the architecture of our feature detection layer in Figure 5. This layer processes on top of the information of the captured images in real time. Our implemented solution involves two different Algorithms detecting two different interaction types. The thumb on hand interaction (c) and the pinching gesture interaction (b). A preprocessor (a) detects the user's hand and the desired gesture type. The resulting data is forwarded to the corresponding feature detection algorithm.

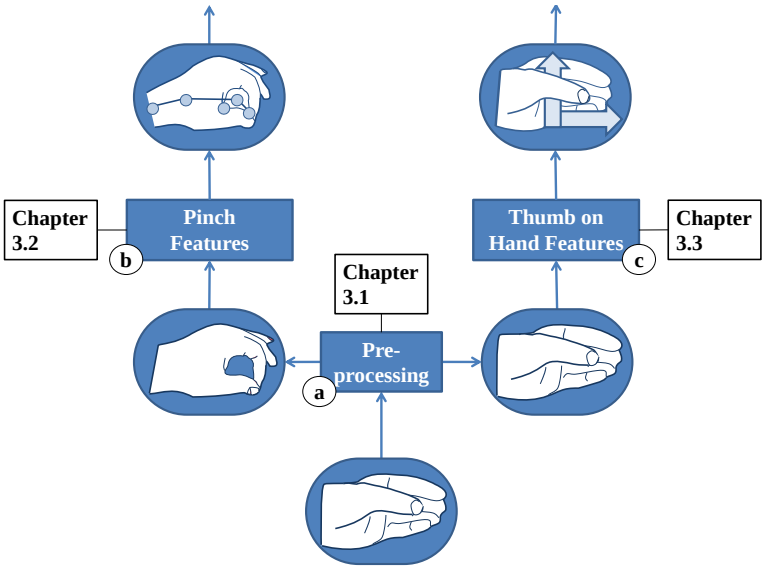

Figure 5: Two different tracking algorithms: We implemented thumb on hand interaction and pinching gesture interaction.

Detecting the Features of the Pinching Gesture

We successfully expanded common techniques to detect pinching gestures by the use of optical systems. In order to enrich the possibilities of interactions, our solution offers a detailed analyse of the user's thumb and index finger as they interact with each other.

Detecting the Features of Thumb on Hand interaction

We implemented the detection of the thumb's relative position in reference to the hand. The finger placed closely together span an input surface the thumb tip operates upon. Hence our solution interprets the hand and its dimensions as a single touchpad. We detect the thumb's relative position and distance to this pad.

3.2.2 *Applications controlled by Imaginary Widgets*

We implemented imaginary widgets, which translate the features of the user's gestures into application input commands. We defined widgets like sliders dials and buttons, the user imagines to be placed along his fingers and thumb. The interaction with those widgets, allows the user to perform continuous and discrete input for numerous type of applications. We designed different visual and audio applications in order to demonstrate the efficiency of our gesture detection technology. A sample Application and the corresponding interaction is illustrated in Figure 6.

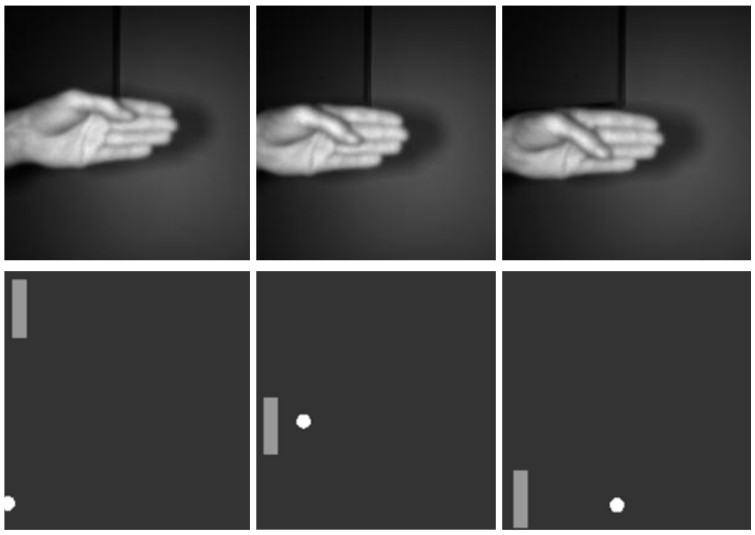

Figure 6: Pong controlled by imaginary slider: The user controls a paddle in order to hit a flying ball, as he slides his thumb tip across the hand.

Since our solution includes two different interaction technologies, we implemented two created two different groups of Widgets. Widgets controlled by pinching gestures and Widgets controlled by thumb on hand interaction. The following paragraphs illustrate the corresponding types.

Widgets based on Pinching Gestures

We designed and implemented the following interaction technologies as illustrated in Figure 7.

- (a) Continous slider along our thumb to be reached by the index finger.
- (b) Continous dial between thumb and finger as they rub each other.
- (c) Discrete itembar along our thumb to be reached by the index finger.

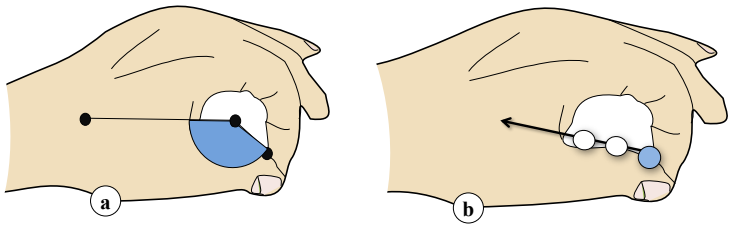

Figure 7: Imaginary widgets based on pinching gestures

Widgets based on Thumb on Hand Interaction

Our imaginary widgets addressing thumb on hand interaction, enable the possibility to use the following types of gestures in order to control different application.

- (a) Touchpad analogy, horizontal and vertical Sliders across our palm.
- (b) dial across the finger.
- (c) Buttons on top of the finger.
- (d) Buttons on top of each fingertip to be reached by the thumb.

By the help of those control elements we are able to build rich applications, which can be controlled by hand and thumb interaction.

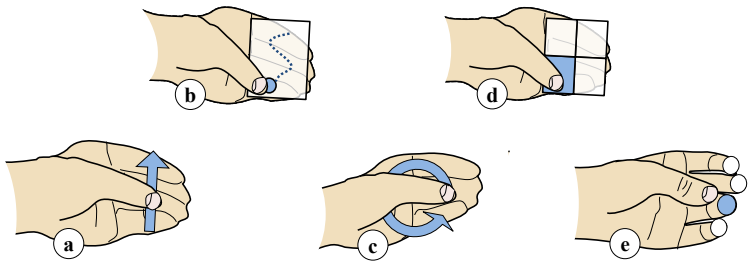

Figure 8: Imaginary widgets based on thumb on hand interaction

Imaginary buttons placed on our palm are implemented in order to control a playlist of an audioplayer or two select different items within a game. Sliders and dials are used in order to change volume or draw lines. The presented widgets enable countless interaction possibilities for many common needs in mobile interaction. Giving the user the possibility to interact with his bare hand, frees him from the need to have a physical device as input space. Eyes free interaction is enabled as we make advantage of the humans ability of receiving tactile feedback, as the thumb tip contacts the hand. In comparison to many gestures involving big movements by arms and body, our desired inputspace is relatively small. This leads to less muscles beeing involved and less public attention beeing caused into our interaction movements and therefor less fatique effects on the user while operating.

3.3 STRUCTURE

This thesis is structured as followed. Our design decisions and implementation are inspired by a great number of prior work. In chapter 4 we describe the corresponding investigations. We illustrate our solution to detect individual features of the user's gesturing hand in chapter 5. The resulting data offer the possibility to define imaginary widgets on top of the gestures, controlling different applications. The relevant procedure is described in Chapter 6. In order to envision and realize a gesture interaction technology we need to narrow down the type of gestures and the detection mechanisms, which are appropriate for our purpose. This assessment is discussed in chapter 7. Finally we conclude the results of our work and encourage future work in chapter 8.

4

RELATED WORK

The related work of this thesis is divided into three main chapter. First of all we want to list work, that motivated us and investigated the need for innovative interaction interfaces for mobile devices. The following section consists of publications that inspired the concept and implementation of our solution for gestural interaction. The last section of this chapter lists commercial and scientific devices that are related to our work.

4.1 INVESTIGATIONS OF MOBILE INTERACTIONS AND GESTURES

Ni and Baudisch [17] illustrated the miniaturization of mobile devices. The ubiquitous use of such devices can be enabled by invisible integration into cloth or skin. This work discusses different sensor implementations and how users interact with them in order to identify boundaries for the miniaturization of mobile devices. The fat finger problem restricts this ongoing process. Therefore the removal of input hardware that is linked to the users finger is recommended.

Gustafson, Bierwirth and Baudisch [9] discovered the humans abilities to interact with interfaces, the user imagines and interacts with. Hence they introduced imaginary widgets the user can take control of, in order to bring gestural interaction to screen-less devices. The input concept consists of a non-dominant hand forming an L-Shape in order to fix a refernce system into mid air and a dominant hand, which interacts within the created space. The corresponding implementation is based upon a chest mounted infrared camera observing the bimanual interaction. Further this work contains an interesting user study, investigating the user's ability to interact within an imaginary space.

Patel and KientzFarther [19] investigated the proximity of users to their mobile phones. The most central idea of this work is the research about users and how their phone is accessible to them. A study including sixteen participants over three weeks is presented. This study showed that users expectation about how easy they access their phone diverge from reality, and depends on certain design factors of the phone. The paper is emphasise the need to develop mobile devices which offer more efficient access.

Cui et al. [5] presented an empirical study of users carrying and interacting with their mobile phone. The data is collected across cultural barriers in four different continents. This research gives insight about personalization, security and different preferences for mobile phones to be placed along the body. Two different types of carrying a phone were identified. On the one side instrumental attributes present: easy to access and secure. On the other side non instrumental attributes: style concentrated and personalized.

Thad E. Starner et al. [22] studied users behaviours and preferences for scheduling appointments. The study includes data from one hundred and thirty-eight participants and concentrates on interaction with PDA's in order to perform the desired scheduling task. PDA users overstated the efficiency of their device. Further this study revealed that a lot of users tend to "buffer" their appointment notes on paper notes for later input to the PDA. This behaviour is justified in the lack of reachability of the device and therefore exaggerates the need to create input technologies that are always at hand.

Kristoffersen et al. [15] explored interfaces for hand held computer supported cooperative work. Empirical research has founded the need for new innovative interaction design for such devices, since the current ones demand to much attention for mobile scenarios. The system called Motil, as introduced in this work, fulfils those demands by limiting visual feedback and enabling audio feedback.

Wolf et al. [25] investigated the ability of users interacting with a device by the use of hand drawn gestures. In order to perform the gesture the user is equipped with a stylus. The main issues of this study are the abilities of users to remember gestures, perform gestures and how much variety users create while trying to perform the same gestures.

Kallio et al. [11] studied technologies and the necessity visualization of gestural commands. This kind of visualization is appropriate especially because of the lack of feedback. Basic concepts for visual animations for gestural interactions are discussed

Kela et al. [12] propose gestural control detected by accelerometer sensors as an alternative to modalities such as styles based interaction and speech recognition. The study found that users find gestural commands more naturally. Additionally the necessity of personalizing commands has been found, since the study proved that different users tend to define different gestures for the same operation.

4.2 CONCEPTUAL WORK RELATED TO GESTURAL INTERACTION

Baudel et al. [1] investigated free hand gestures using a DataGlove. This device can measure the orientation of the hand and position of each finger. It is used to control a presentation on a remote computer.

Bowman et al. [2] illustrated a menu interaction system based on bimanual interaction by the use of Pinchgloves. This cloth glove can detect pinch gestures and contact between each finger due to electrical sensors weaved into the glove on each fingertip.The Tulip Menu enables the possibility to access top level menu items by pinching fingers to the thumb on the non dominant hand and menu commands by pinching fingers on the dominant hand. A study showed that the presented system is more comfortable and became efficient after a short learning phase compared to two common Virtual environment

systems.

Ni et al. [18] presented the Rap Menu. A system to navigate through menu items by rotate and pinching. The user rotates his hand in order to reach certain areas of a radial menu layout. By pinching individual Fingers he is able to select the desired item.

Kohli et al. [13] investigated bimanual interaction in virtual environments. The dominant hand interacts on the non dominant hand. A user study showed that using physical props for tactile feedback, helps users to interact in virtual environments more effectively.

Wilson [24] illustrated a lightweight computer vision algorithm, detecting when a user brings index finger and thumb together. This pinch gesture can be detected by the use of connected components. The hand is detected as a closed connected component, as the static background is erased due to background subtraction . The pinching gesture is identified, when thumb and index finger form a hole within this connected component from overhead view. The pinch itself as discrete input and its position relative to the camera as continuous input, is used in order to control a desktop scenario and a map application.

Ashbrook,Starner et al. [7] presented the gesture pendant input system for home automation concepts. Gestures are detected by an infrared camera as a necklace mounted to the chest. The camera is ringed with infrared Led's flooding the area in front of the users chest with infrared light. Enabling gestural interaction, offers interfaces to entertainment systems, lightning and heater and is especially appropriate for users with limited sight or motor abilities. Additionally the Gesture pendant can be used as a health monitoring device, due to the fact that the target group of the elderly and disable people often suffer from diseases which have tremor symptoms. Those tremors can be detected by the gesture pendant and evoke an alarm. User defined Gestures and Control Gestures are identified by computer

vision and Hidden Markov Model.

Starner et al. [21] investigated a System to detect sign language by the use of camera tracking. Hidden Markov models are used in order to determine he identity of the performed gesture. Two different designs are used in order to set the view port of the camera. A desk mounted camera for static scenarios and a cap mounted camera for mobile scenarios. The desk mounted system achieves an accuracy of ninety-two per cent. The cap mounted reaches up to ninety-seven per cent.

Harrison et al. [10] presented the skinput system. A system in which the skin along the forearm is used as input space. Contact of the Fingertips on the non dominant arm are detected by acoustic sensors. Those sensors can measure the mechanical vibration of the skin evoked by a tap. Additionally the system is equipped with a projector displaying different menu items along the input area for visual feedback.

Bretzner el al. [3] presented computer vision algorithms to detect hands in image frames and determine their position and pose. The algorithms include particle filter and multi-scale color feature detection. A home automation system can be controlled by the use of the defined gestures. Early studies showed that their current approach suffered from the users fatigue and ability to control marking menus by the use of gestures.

4.3 GESTURAL INTERACTION DEVICES

Kolb et al. [14] gave a very interesting overview of depth cameras equipped with the Time-Of-Flight Technology. It consists of descriptions about different TOF technologies, their advantages and drawbacks, different noise types and the corresponding algorithmical solutions representing the state of the art of image processing. This

work is highly recommended for everyone, attending to work with TOF cameras.

Zimmerman et al. [26] discovered early in 1986 the use of an electronic glove as an interface device. The Glove enables the ability to measure hand orientation, bending and position of individual fingers. Haptic Feedback for bending different fingers and vibrations are also provided by those gloves. The Z-Glove and DataGlove are used in order to manipulate virtual objects.

PowerGlove [6] is a commercial product by mattel released in 1989 for the nintendo entertainment system. The glove can measure yaw and roll of the users hand. Build-in flex sensors measure the bending of the finger. Additionally buttons are placed on the back of the glove.

The P5 glove [8] is designed for human computer interaction. The system can detect the gloves yaw,pitch and roll by a low degree of noise (1 degree). An optical tracking system determines the gloves position in a range up to four foot from the camera. Additionally the fingers are equipped with sensors, enabling the detection of the fingers bending by an accuracy of 0.5 degree.

The Wiimote [4] is an input interface for the gaming console Nintendo Wii and was released in 2006. It allows gestural input by accelerometers, measuring the Hands orientation and acceleration. Additionally the device is equipped with an infrared camera, that tracks sensor bar LED'S placed next to the TV. This technology offered game designers the ability to develop pointing and shooting game concepts for the Wii-console. The data collected by the Wiimote is transferred via bluetooth to the gaming device in 100 data packages per second. The hardware plug-in Wiimotion Plus, consisting of a dual axis gyroscope and a single axis gyroscope, offered higher accuracy for motion detection.

Playstation move [16] is a motion detecting device developed for Playstation in 2010. It consists of a light ball filled with RGB light-emitting diodes on top of the device. This Ball can be tracked in three

dimensional space by the Playstation camera which is mounted to a static object close to the television. Additionally the device contains several sensors to determine the devices rotation and a magnetometer for calibration.

Microsoft's kinect camera system MicrosoftKinect:10 calculates three dimensional information of the observed area based on structured light technology. It operates in range of 0.7 meter up to 6 meter. A structured infrared pattern is projected onto the scene. The deformation of the structure observed by the infrared camera can be used in order to recreate the three dimensional structure of the observed area. The ability to separate the users body from the background and form a generic skeleton, enables motion tracking.

5

IMPLEMENTATION

The following chapter presents our implementation of thumb and hand interaction. The corresponding dataflow, the individual components and their architecture is illustrated in Figure 9

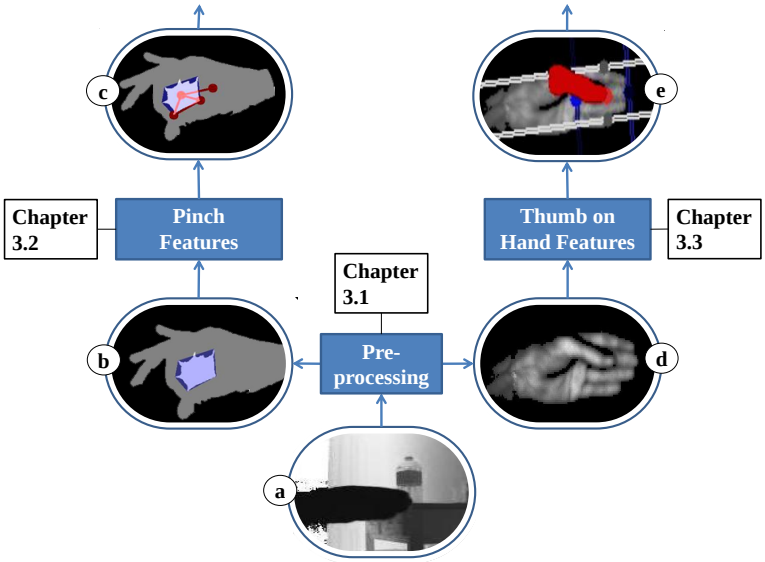

Figure 9: Overview of feature detection: The preprocessing component extracts the user's hand from the background. The isolated mask of the hand is used in order to analyse the features of the interaction.

The design of our mobile interaction technology involves a camera, which continuously records the area in front of the user's body. Hence the implementation of our gesture detection demands a high quality of avoidance of unwanted activation by irrelevant data. A robust separation, between dispensable background data of our changing environment and the user's hand gestures is a crucial task. A preprocessing unit (a) fulfils this demand, by continuously preventing useless data from being parsed to our interaction algorithm. We implemented two different algorithms tracking two different types of interactions. The group of pinch like interactions (c) and the group of thumb on hand interactions (b). The feature analysis of the

user's pinch gestures is presented in Section 5.2. In Section 5.3 we discuss our technique for detecting the thumb on hand interaction. The following paragraph introduces a high level overview of foundational steps in processing image data. The reader of this thesis is assumed to be aware aware of the following basic concepts and introduced terms.

OVERVIEW OF COMPUTER VISION BASED FEATURE EXTRACTION

This section discuss the foundation of processing image data by the help of Open Computer Vision and illustrates our general course of action. OpenCV is an open source library, which consists of procedures for managing image data. Image Data is stored by a raster image defined by a m*n Matrix. OpenCV offers the possibility to process on top of this data with statistical procedures in real time.The following list of features determine the main advantages of OpenCV:

- Motion analysis
- Image segmentation
- Image smoothing
- Feature detection
- Histogram analysis
- Principal component analysis
- Geometrical functions

Our general workflow for feature detection in image data is illustrated in Figure 10.

- (a) The first step is to capture the image frame.
- (b) Based on prior information we create a mask of interest.
- (c) The mask is interpreted as a connected component.

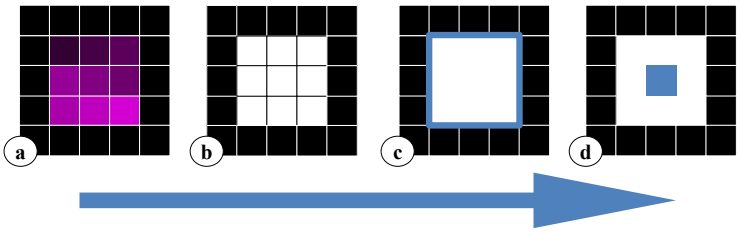

Figure 10: Overview of processing image date: Capture the image(a), Mask of interest(b), Connected components (c), feature detection (d)

- (d) We detect features of our connected component. The centroid of our connected component is illustrated as a sample feature.

Capturing image frames

Each entry of the image contains information, representing the intensity of the attribute. Within a grayscale image, a single sample attribute is stored in a single channel, defining the amount of light varying from black, across different gray shades to white. This becomes helpful if we want to measure a single band within the electro magnetic spectrum, for example the intensity of infrared light that captured objects reflect. Our solution takes advantage of this type of matrix in order to store two type of informations captured by our camera. Grayscale images containing the intensity of infrared light measured and distance values for each pixel, referencing the distance to the lens.

In color images three channels per pixel offer the possibility to store three samples, defining the color of the pixel. We use this type of image in order to store the values of the captured three dimensional environment. In this case RGB values are replaced by the x,y and z values of the real world's coordinates.

The following Figure 11 illustrates the types of images, relevant for our solution.

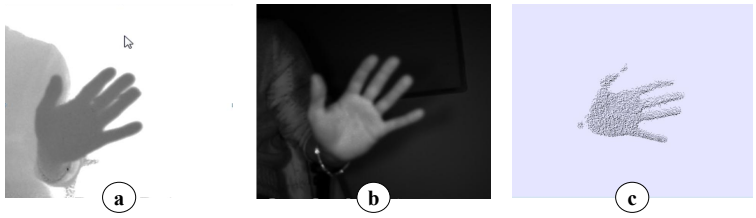

Figure 11: The different types of images: The distance image (a) relations the distance to the lens for each pixel. The infrared greyscale image (b) contains the infrared measurements. Three-dimensional information visualized as a point-cloud (c) in OpenGL

The Region of interest

The region of interest within our image contains the data, which is relevant for further analysis. Useless data needs to be reset to zero in order to avoid further proceeding to critical Sections. In general this procedure is performed by thresholding. The definition of a lower boundary and an upper boundary excludes irrelevant data from further processing and generates a spectrum of values that are valid for the region of interest. In case of depth images, thresholding is performed by setting upper and lower boundaries for the distance values of each entry within our distance image. Additionally further constraints need to be implemented in order to identify the area that is desired to detect.

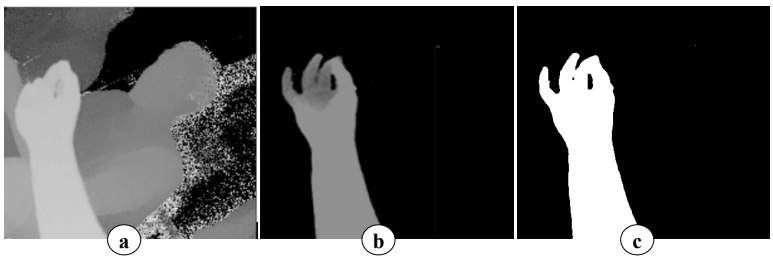

Figure 12: The Region of interest

Figure 12 illustrates the process of thresholding an image to our region of interest.The raw distance image contains all of the captured information (a). On top of this raw data,we created a mask

only containing the data of our interest (b). This mask can also be transformed to a monochromatic image (c) only containing binary information about. This binary mask can be used in order to map it to other images, for image comparison or in order to calculate connected components.

The Connected Components

Connected components are areas of neighboured pixels which are equally labeled within a binary image. Each connected area of pixels represents its own connected component. OpenCV offers the possibility to detect connected components and store their encasing contours within a tree structure, representing topology and hierarchy of our mask. The mechanisms to label the assignment of each component are illustrated in Figure 13. The background area is identified as invalid and is not assigned to the component's storage tree. The main component is the biggest connected area of valid pixels. The enclosed component is stored as a child node of the main component.

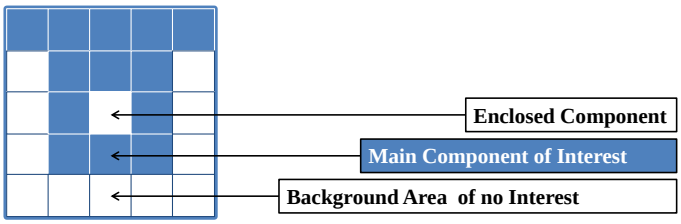

Figure 13: The concept of connected components: The enclosed component is identified as a hole within its host component.

Features

Connected components offer the possibility to detect different features of the mask. The following Figure 14 illustrates some of the most important features, that are used within our solution and operate on top of connected component data.

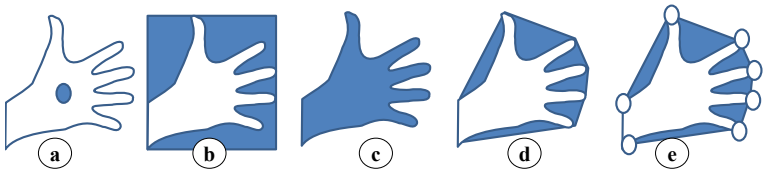

Figure 14: Features: Center of gravity (a), bounding box(b), geometrical area(c), hull(d), convexity defects(e).

- (a) Center of gravity is the mean of all the mass within the image. The advantage of working on top of this feature is, that it is relatively stable and fixed in relation to the orientation of the shape. Small groups of outliers around the shape, appearing due to measurement noise, do not effect the position of the center of gravity in a significant manner.
- (b) The bounding box function is a fast and robust way of identifying the region of interest within our image. It returns the smallest upright rectangle containing all of the mask's information.
- (c) Geometrical functions offer the possibility to calculates the size of the area.
- (d) The convex hull function calculates the convexity of a 2d set of points using the Sklansky [20] algorithm.
- (e) Convexity defects are outstanding areas of the mask disturbing the continuity of the corresponding convex hull. This function is commonly used in gesture detection in order to identify regions, which indicate potential fingers segments.

Constitutive Feature detection

This paragraph illustrates the use of foundational features within an interplay in order to detect advanced features. The following example of detecting fingertips, its underlying mechanisms and terms are reused within the upcoming implementation Sections.

We propose the usage of a virtual field of view in order to locate the region where we expect the fingertips to arise as illustrated in Figure 15. The calculation of a virtual field of view is a fundamental procedure within our solution. Based on a direction vector, a position vector and an angle of view, we are able to fix this field of view in two dimensional space and define a subregion of interest within our mask. We start by calculating the point where our hand intersects with the edge of the frame. This point is called the enterpoint and can be determined by iterating through the pixels of our mask, searching for all pixels which intersect with the frame's edge and calculating their mean. The orientation vector is spanned from the enterpoint to the spatial moment. The direction vector of our virtual field of view is now defined by the calculation of our hand's orientation. We use the spatial moment as the position vector and fix the field to this position. Our investigations found an angle of view of 110 degree as appropriate to cover all of the fingers within the field of view.

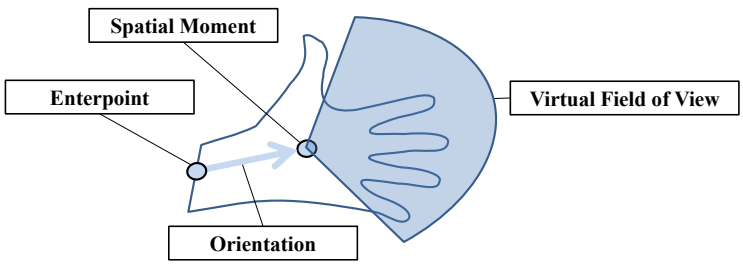

Figure 15: The virtual field of view: fixed to the spatial moment, it covers an area of interest

We continue to calculate the convexity defects laying within the virtual field of view as illustrated in Figure 16. This function, as implemented in OpenCV, offers the possibility to store convexity defects as a list. Each single sample is represented by a tuple

(Start,End,Depthpoint,Depth). The start and end points define the fingertips. Depthpoints represent the space between the fingers. The values of those attributes can be used for further investigations, concerning the individual fingers or gestures.

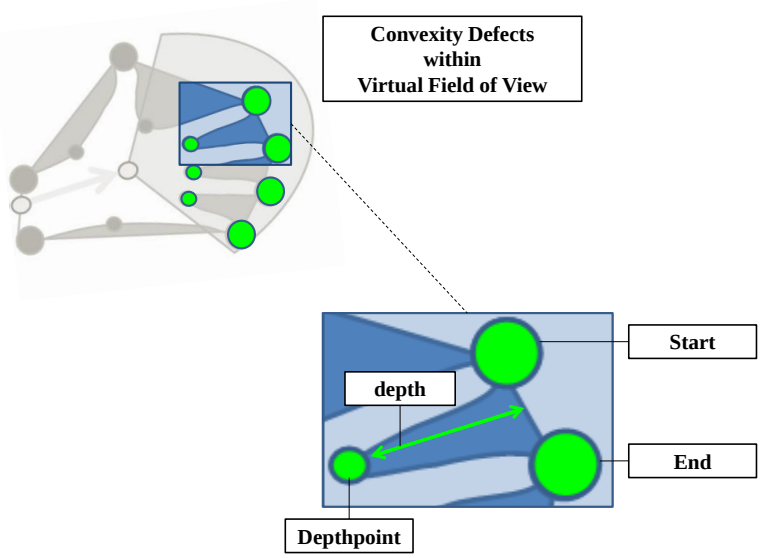

Figure 16: Detection of fingers by the use of the virtual field of view: The virtual field of view is fixed to the spatial moment, it contains an angle of 110 degree and covers the finger

5.1 PREPROCESSING THE IMAGE

Our system needs to be able to identify the data of the user's gesturing hand even in mobile scenarios. Hence it is crucial to avoid unwanted activation of our mobile device, by preventing irrelevant data from being processed by the interaction detection solution. Therefore we implemented three different components, forwarding only relevant data of the user's hand to the corresponding tracking algorithm. The concept of our preprocessing solution is illustrated in Figure 17.

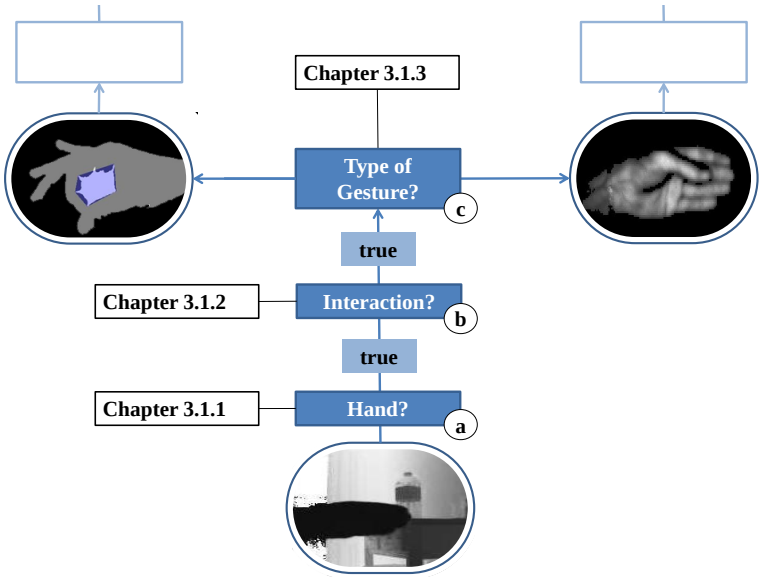

Figure 17: Overview of the preprocessing

- (a) The first component analyses spatial properties of observed objects, in order to separate the user's hand from the background.
- (b) The second component processes on data which successfully passed the first component. It analyses the hand's motion and shape in order to detect if the user performs the system's activation gesture and its state. This component can be understood as an additional barrier detecting if the detected object is a human hand. We found random objects and hand's to quite unlikely simulate the defined key gesture by accident.

- (c) The third component analyses the hand, in order to detect the user's desired gesture and proceed the data to the corresponding algorithm.

5.1.1 *Detecting the Hand*

In this Section we want to present our light weight algorithm, detecting if the captured image contains data of a potential hand. Identifying objects in a camera image is a complex problem, motivated by multiple research fields. Generic solutions suitable for universal conditions are still futuristic visions. In general it is appropriate to tie the algorithm to the conditions of the camera system. If we are aware of the properties of our camera and the physical features of our object of interest, then we are be able to define logical assumptions of the expected data corresponding to the desired object. Our Camera is mounted to the chest and has a relatively small angle of view of 40 degree. The user's interacting hand has to be placed within this angle and shouldn't and is not expected to be further away then a meter. This physical precondition enables the possibility of defining two types of features as listed below.

- The Distance: Subtracting the background based on depth information.
- The Spatial properties: the two dimensional mask of the hand occupy characteristic features.

The corresponding algorithms are discussed in the following paragraphs.

The Distance

Our device supports the ability to capture depth information of an area within the range of 0.3 up to 7.5 meter in distance to the camera. Because of natural restrictions of the users arm length, we assume the hand being within one meter in distance to the camera. Therefore

the space of interaction is within a distance range of 0.3 and 1 meter. We identify close objects by thresholding the distance image to the corresponding boundaries as illustrated in Figure 18.

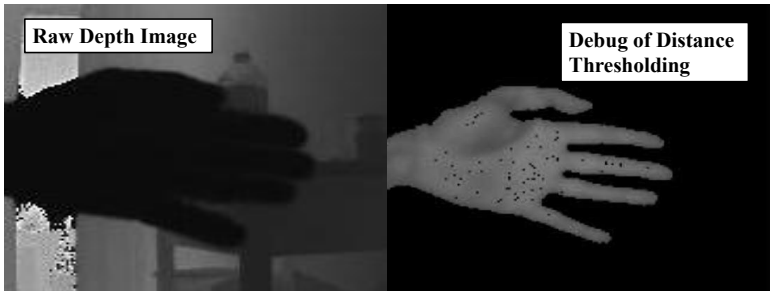

Figure 18: Thresholding the depth image: The raw image displays multiple multiple objects within our interactionspace. The thresholded image illustrates the user's hand extracted from the background.

The Spatial Properties

Once potential objects of interest are identified based on depth data, we proceed by analysing the two dimensional spatial properties of the corresponding masks. Both of the following constraints need to be fulfilled for further processing and are illustrated in Figure 19.

- (a) The user's hand is supposed to be the closest object to the camera. Hence we assume the hand contour to be the biggest connected component in the contour storage tree.
- (b) The user's hand is supposed to enter sidewise the field of view of our camera. Therefore our desired connected component needs to cut the edge of the frame. In order to monitor this feature, check for the existence of the Enterpoint.

Once the listed constraints are fulfilled we forward the data of the potential hand mask to the next component described in the following subsection 5.1.2

5.1.2 *Monitoring the state of interaction*

This subsection describes the implementation of an activation gesture, unlocking the system to read the user's gestures. We implemented

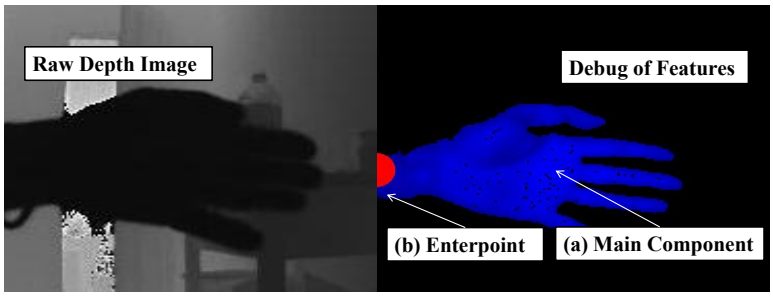

Figure 19: The Analysis of spatial properties: The thresholding processing on the raw depth image enable the possibility to detect the main contour. This contour contains the biggest area and enters at the edge of the frame

the following activation Gesture as illustrated in Figure 20. The interaction system remains in passive state as long as the illustrated sequence of signs is not performed in its defined row and rhythm. In order to deactivate the system, the user spreads his fingers again as he did in the activation gesture (Phase 2) or moves his hand out of the frame.

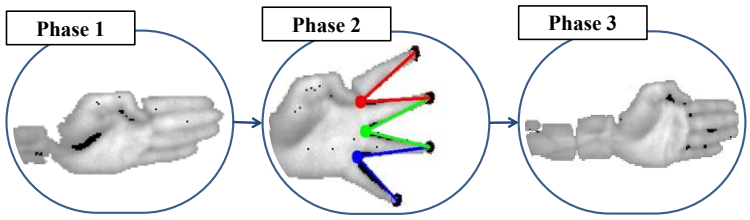

Figure 20: The activation gesture: The Sequence of signs is divided into three phases. A closed hand (Phase 1), an open hand (Phase 2) and again a closed hand (Phase 3)

The following paragraphs illustrate the constraints of each individual state, which need to be fulfilled in its corresponding order.

Phase 1 Hand registration

The first step within our sequence analyses the data directly parsed from the prior hand detection component. The object interpreted as a potential hand, has to fulfil all of the following constraints in order to be successfully detected.

- Motion: We continuously calculate the masks's spatial moment and compare it with the previous measurement. This lightweight motion analysis offers the possibility to detect a rough understanding of the speed of the hands movement. Only if the hand moves slower then a pre defined upper boundary, the sign becomes labelled as true. This procedure is needed in order to exclude fast irrelevant gestures the user performs in front of his body.

- Rhythm: A time stamp is measured once the hand enters the frame and fulfils the Motion constraint. The sign is labelled as detected, if the mask remains within the frame longer than a half second and no longer then two seconds. Hence only within this intervall the data of the mask is allowed to proceed into Phase 2.

Phase 2 Revealing Finger

The second step is the most crucial within the sequence, because the user reveals a number of natural features of his hand as he spreads his fingers. We are able to detect those fingers by calculating convexity defects as illustrated in Figure 21.

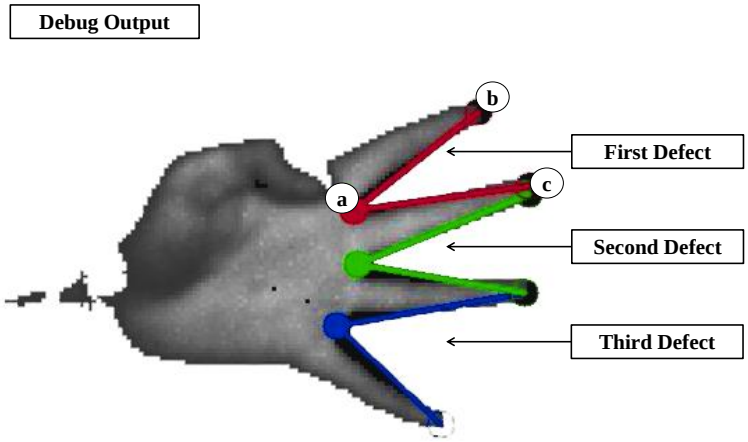

Figure 21: The convexity defects of the user's finger: Each individual defect is illustrated by its coloured vectors

We detect three defects identifying four fingers. We use the tuples returned by the convexity defect function of OpenCV, in order to monitor the following constraints.

- Angle of Finger: The angle of spread of each finger to its neighboured finger has to be bigger then 10 degree and smaller then 60 degree. This constraint fits to the natural conditions of the human hand. We calculate the angle by defining two vectors for each defect. One vector reaches from the depthpoint to the startpoint, the other from the depthpoint to the endpoint. The angle in between the two vectors represent the angle of spread of each finger.

- Size in 2D Space: The depth of each defect has to be longer then 3 pixel.

- Size in 3D Space: The average length of all of the finger in three dimensional space, has to be longer then 3 cm and shorter then 13 cm. The implementation of this constraint is illustrated in Figure 22. We identify the position in three dimensional space for the depthpoints, startpoints and endpoints for every single defect. The length is measured for each of the three depthpoints to its individual start and endpoint. The average length of the finger is defined as the mean of the resulting six direction vectors. Calculating the mean is necessary since the finger contain a significant amount of noise around the fingertips. Additionally the result of this calculation is reused in Section 5.3.

- Motion: The hand is supposed to not move in significant manner. We monitor the motion as described in Phase 1.

- Rhythm: The sign is labelled as detected, if the above mentioned constraints are true longer then a half second and no longer then two seconds. Within this span the data of the mask is allowed to proceed into Phase 3.

Phase 3 Completing the Gesture

The final gesture is a closed hand. This final state of our sequence is simply limited to the constraint of not being allowed to move abruptly through the images frame. The corresponding motion is supervised as described in the previous phases of the sequence.

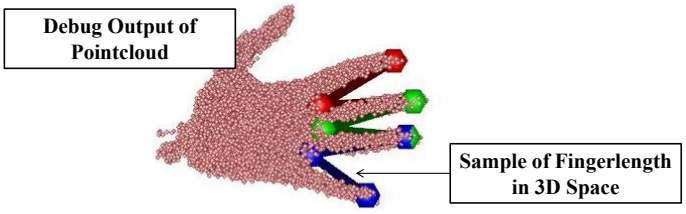

Figure 22: Measurement of the finger's length in three dimensional space

5.1.3 *Detecting the Type of Gesture*

This Section describes our solution to detect the user's desired type of input as illustrated in Figure 23. The user unlocks the device by the activation gesture, which is described in the previous subsection 5.1.3. Once this activation gesture is within its third and final phase, the data of the hand's mask is forwarded to this component, detecting the type of interaction. If the user performs a single pinch gesture within an initial time slice of one second, the data of the hand's mask will be continuously parsed to the component detecting the features of the pinch. If this gesture is not detected, data is forwarded to the component, detecting the features of thumb on hand interaction. The result of this conditional construct is constant as long as the individual interaction stays in its active mode. It is reset as soon as the state of interaction turns passive. The relevant algorithms in order to robustly detect if the user performed a pinch gesture are discussed in Section 5.2.1

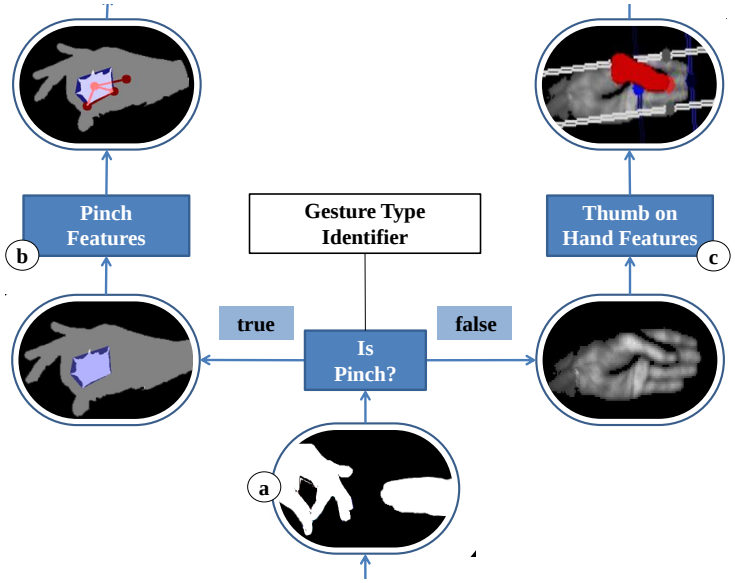

Figure 23: The gesture identification: The hand's mask (a) is initially checked for the existence of a pinching gesture. Depending on the conditional result, the data is forwarded to the component detecting the features of the pinch or to the component detecting features of thumb on hand interaction.

5.2 DETECTION OF PINCH

The contour of the pinch is constructed by a hole between thumb and index finger as they contact each other. The following chapter presents our solution to robustly detect the existence of the pinch and its features. The corresponding components and their position within our architecture is illustrated in Figure 24. The algorithm (a) for the detection of this area is described in Section 5.2.1. The detailed analysis of the pinch component (b) is presented in Section 5.2.2.

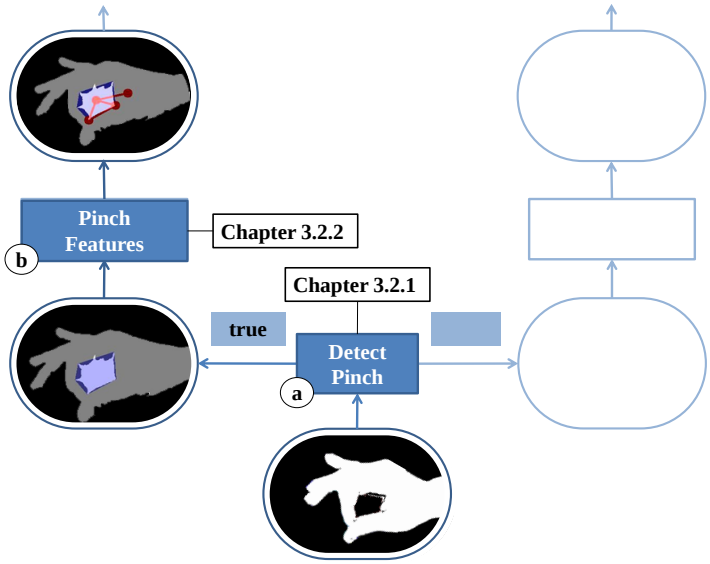

Figure 24: Tracking of pinch interaction

5.2.1 *The Pinchmask*

The upcoming subsection describes our solution to detect the pinching gesture. Since our algorithm is based on the content of the two dimensional mask and its components, we can simply abandon the concrete depth and gray information of the image. Shadow effects, cloth or jewellery can create noisy measurements. In order to avoid those effects arising wrong detection, we need to get rid of those areas by simply dilating the image. The calculation of the connected component of the hand's mask as illustrated in Figure 25, offers the possibility to determine its spatial properties. The area (a) contains useless background information. The child node (c) within the hand's component (b) represents the desired shape of the pinch.

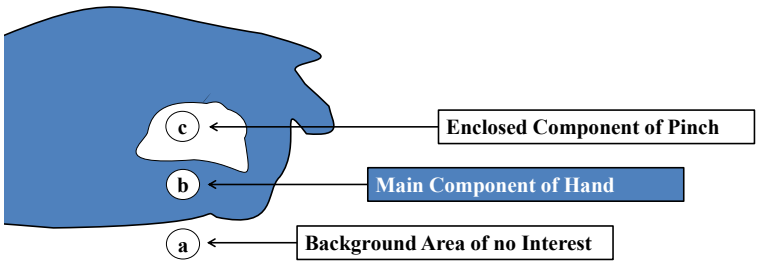

Figure 25: The connected components of the mask: the area (a) contains useless information. The pinch (c) is a child node of our hand's component (b).

After we detected the existence of the pinch component,we need to ensure that this area is created by the desired gesture, instead of noisy fragments. Hence we propose to check if the position and size of the pinch component is reasonable in reference to our hand's orientation. As the user reaches sidewise into our field of interaction, we assume the pinch component to appear roughly within the end of our forearm. Additionally the component of the pinch must contain a significant size in relation to the hand's mask. We reuse the concept of the virtual field of view as illustrated in Figure 26, in order to monitor our assumption concerning the position of the pinch. The area of the pinch component must lay within the virtual field of view and contain at least a twentieth of the area, which is occupied by the hand's component.

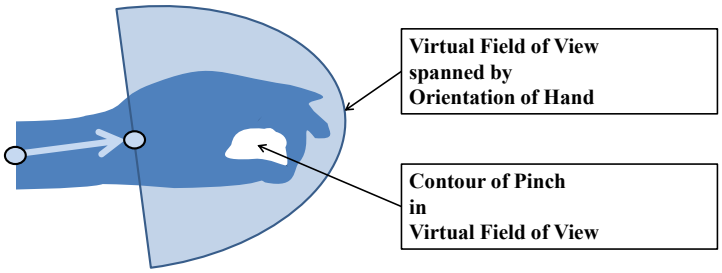

Figure 26: Conditions of pinching gesture. The Pinch must lay within the virtual field of view and contain at least a twentieth of the area.

5.2.2 *Pinch features*

The following subsection discusses the detection of the features of the pinching gesture. The main feature we are looking for, is the point of contact between thumb and index finger and how this point is positioned in relation to a robustly fixed number of general features of the hand. The sum of characteristics, that is needed in order to identify rich interactions, is illustrated in our debug-output image of Figure 27.

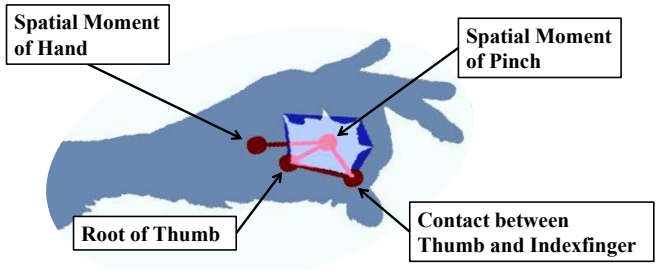

Figure 27: Constuct of pinching features: The interplay of different spatial features enables the possibility to detect detailed information of the interaction

The identification of the thumb is the most foundational task in order to detect the contact between the thumb and the index finger. We simplify this challenge by approximating the contour of the pinch down to a Polygon, which consists of only 4 Vertices as illustrated in Figure 28. The region of the thumb is adjacent to one of the four vectors of the polygon.

In order to determine which Vector fits to the thumb, we need to proceed by generating a more precise understanding of the shape

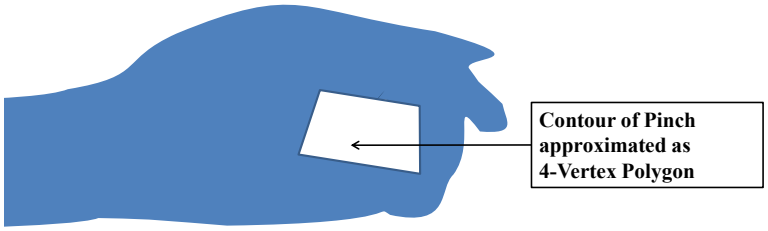

Figure 28: The approximated polygon of the pinch contour

of the hand's mask. We detected a rough orientation of the hand in the previous subsection. For further processing we need a more precise solution, calculating the hand's orientation and excluding the orientation of the forearm. We do so by spanning a vector from the spatial moment of the hand component to the spatial moment of our pinching component.

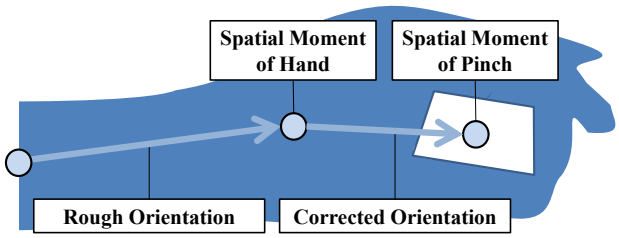

Figure 29: The corrected orientation: is represented by a vector spanned from the spatial moment of the hand to the spatial moment of the pinch

The prior assumption that our hand is held as illustrated in Figure 30, simplifies our task in significant manner. Hence we assume the vector representing the thumb, to lay below the spatial moment of the pinching component. We identify this vector, by spanning an orthogonal vector in reference to the orientation of our hand, which roots down to the bottom of the polygon. This vector intersects with the vector, representing the thumb. The identification of this vector and their corresponding vertices, represent the two most relevant features of our solution. The Position of the point of contact between thumb and index finger and the position of the point where our thumb roots out of the hand

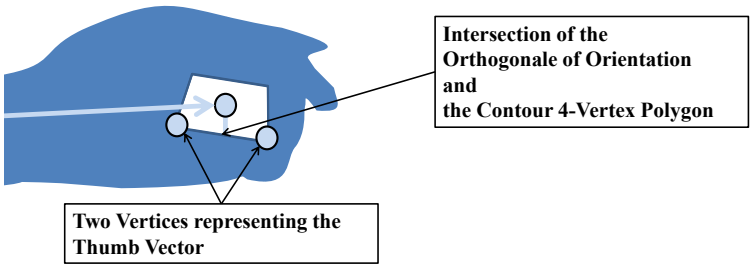

Figure 30: Detecting the vector representing the thumb and its vertices.

The described algorithms compile a construct of features as illustrated in Figure 31. This construct enables the detection of detailed information about the interaction that is performed by the user. We implemented imaginary widgets on top of those features in Section 6.1.

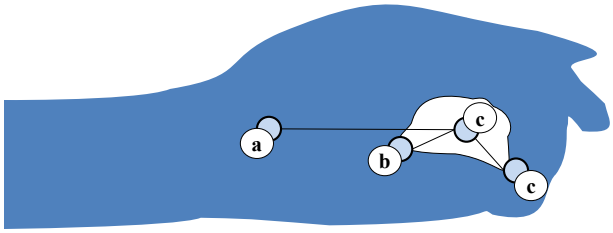

Figure 31: The construct of detected features: The spatial moment of the hand (a) and the pinch (c) in relation to the rooting point (b) of the thumb and the point of contact between index finger and thumb (d).

5.3 DETECTION OF THUMB ON HAND INTERACTION

The following chapter describes our implementation of thumb and hand interaction. Our solution takes advantage of the fact that the thumb represents an exceptional position according to the hand. We approximate the palm of our hand and fingers as a flat surface. Detecting the position of the thumb in reference to this surface, enables the possibility to create thumb and hand interactions as illustrated in Figure 32.

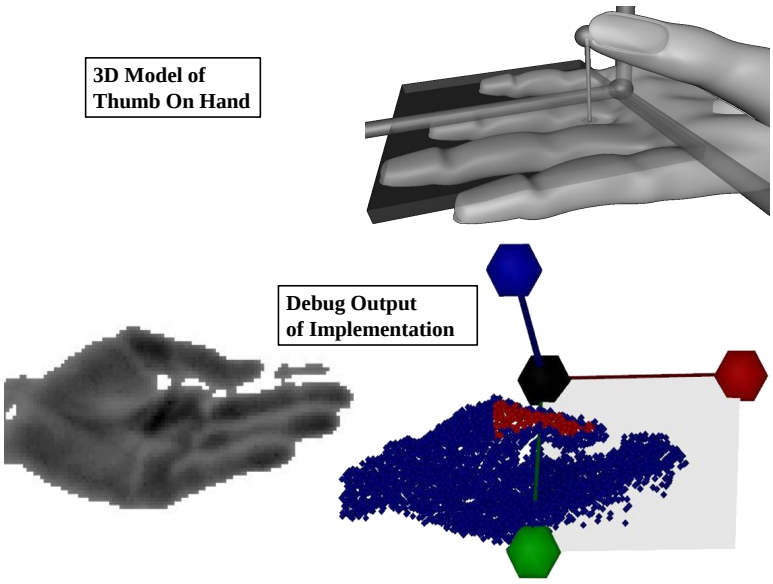

Figure 32: The hand as a touchpad: The position of the thumb in relation to the surface is illustrated as a conceptional model in comparison to our debug output

The architecture of our solution is illustrated in Figure 33. The most foundational step is to smooth the image data and erase wrong measurements before further processing, since further procedures within our solution are very sensitive concerning the correctness of the depth data. After determining foundational features of our hand's orientation and size, we use statistical methods in order to approximate the hand as a flat surface. The resulting three dimensional plane is a useful tool in order to detect the thumb, since it occupies a distanced position in reference to this plane. By the use of the Data of our hand's plane and the thumb's features, we are able to detect

the thumb's relative position in relation to the corresponding hand. Additionally we describe our solution to detect the thumb tapping individual fingers.

The following Figure 33 illustrates the structure of the following section and the component's architecture.

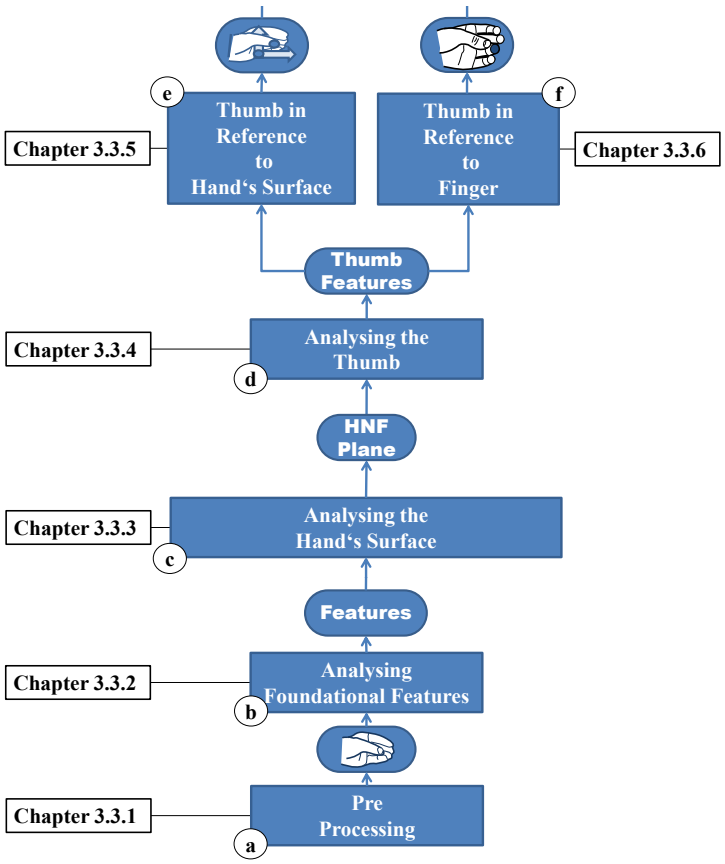

Figure 33: Architecture of the detection of thumb and hand interaction and its corresponding chapters within the thesis

- (a) Preprocessing detects noisy depth measurements within our image
- (b) Analyse of foundational features is needed in order to segment the hand in regions corresponding to their dimensions
- (c) Analyse of the hand's surface
- (d) Analyse of thumb

- (e) Detection of the thumb's relative position in reference to the hand
- (f) Detection of the thumb's relative position in relation to the finger

5.3.1 *Preprocessing the Image*

The following subsection describes our simple light weight procedures, in order to increase the precision of our data. The image data captured by depth cameras is sensitive to depth informations, especially along the depth axis. Hence it is necessary to exclude noisy measurements from beeing parsed to further algorithms. The efficiency of our preprocessing unit in contrast to noisy raw data is illustrated in Figure 34.

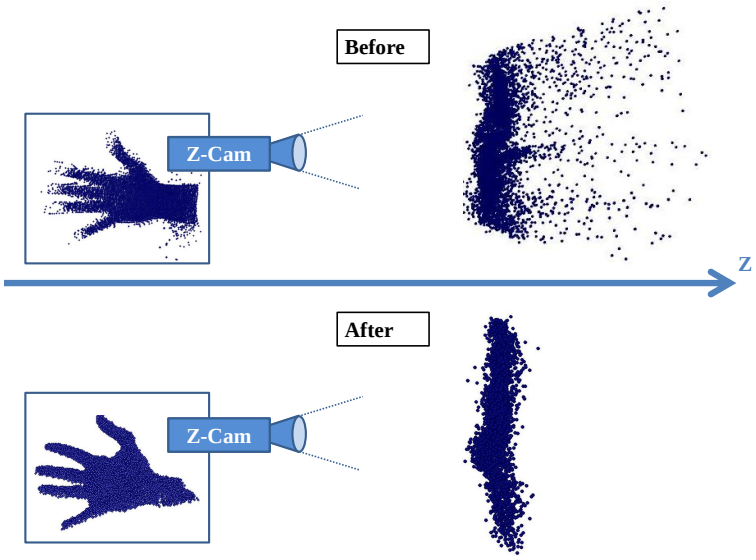

Figure 34: The efficiency of our preprocessing unit: Both images illustrate the corresponding debug output of OpenGl. The hand is recorded from the front. The initial point-cloud, captured by the depth camera, reveals a significant amount of noise along the z-axis. Our preporcessing unit successfully detects outliers and produces a detailed point-cloud.

Detecting outliers within a point-cloud, which is produced by a depth camera, is a complex task. However, since we assume the hand's skin to contain a continuous diffuse reflectivity, we are able to solve this problem by the use of a light weight solution, processing in real time. Statistical outliers within our data will be detected in two different data rooms representing the same mask of interest. The data of our two dimensional infrared mask and the three-dimensional point-cloud. The detection of valid pixels within both images is

implemented in the following two sequential passages and generate one global mask of valid pixels.

Erasing noisy pixels based on depth measurements

Noisy pixels, which contain invalid z-values can be detected by measuring the depth mean and the standard deviation. This is possible since we are only interested in detecting a single object, which doesn't offer any rapid falloffs in depth. Three dimensional entities, which contain a depth value smaller then the half standard deviation subtracted from the mean is labelled as invalid.

Erasing noisy pixels based on infrared amplitude measurements

The user's skin reflect the infrared light emitted by the depth camera system. A significant amount of depth measurement errors appear due to not enough infrared light being reflected by the object. However, since the user's skin is expected to contain a nearly constant amount of albedo, we assume the hand to reflect a constant amount of infrared light. This precondition indicates that noisy measurements within the amplitude image, occur due to the shape of the hand. Round finger's segments and the edge of the hand doesn't reflect enough infrared light in order to determine the distance to the camera. We need to identify those areas of wrong measurements, based upon the standard deviation and mean of the hand's infrared mask. Each entity within our image, which contains a value smaller then the standard deviation subtracted from the mean are labelled as invalid.

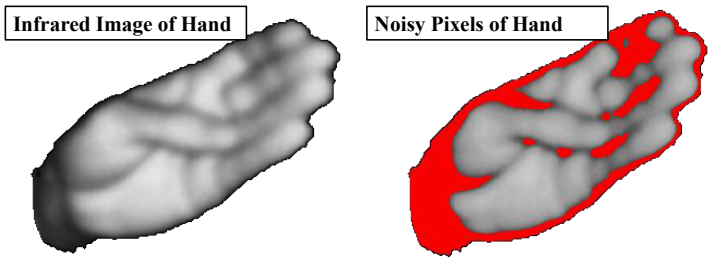

Figure 35: Noise detection based on the infrared image: red coloured areas are labelled as invalid.

The following Figure illustrates different point-clouds, visualized in OpenGL. Each point-cloud represents a state of preprocessing. The raw point-cloud (a) contains a significant amount of noise. Further precise calculations based on top of this data is not appropriate for our purpose. The point-cloud (b) is the resulting point-cloud, after elimination of noisy pixels, detected within the depth data. The result is promising, but still to inaccurate in order to detect the thumb. The quality of our final solution is illustrated by point-cloud (c) and offers the possibility to detect the hand's surface and the thumb.

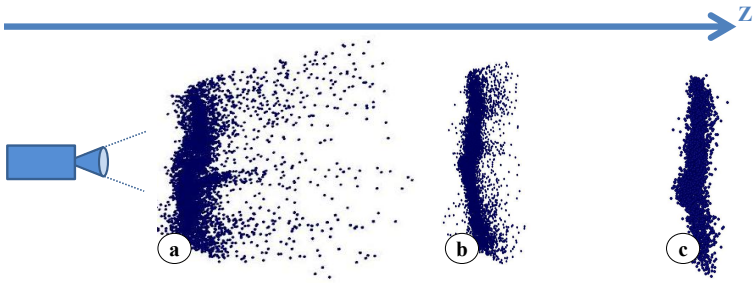

Figure 36: Comparing the different states of noise detection: the point-cloud (a) is a debugging visualization of the raw depth values. point-cloud (b) is the remaining mask after the outlier detection based on depth data. The refinement (c) is realized by the infrared data.

5.3.2 *Detecting fundamental features of the hand*

The main goal of the upcoming subsection is to identify the data representing our fingers. This is a fundamental step in order to detect the interaction between the thumb and the inputspace, which is spanned by the fingers. We present a generic fragmentation of our hand as illustrated in Figure 37.

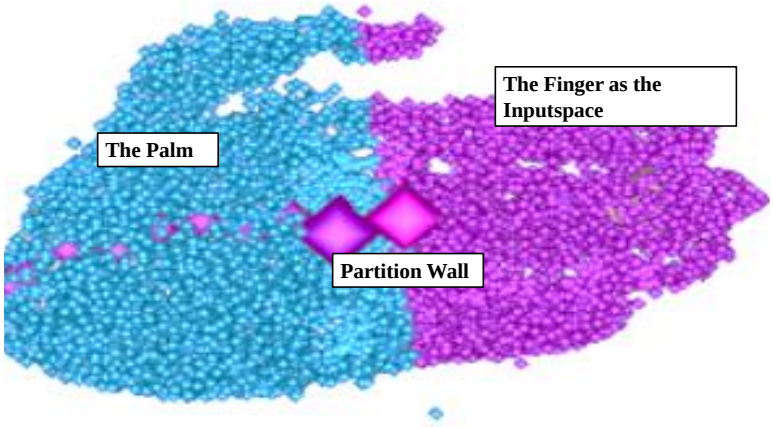

Figure 37: Detecting the inputspace: The Finger are separated from the palm by a partition Wall

The pixels of our finger represent a coherent group of entries in our mask. This essential precondition reduces the complexity of our task in a significant manner. We need to identify the location and the size of the area representing our finger, by detecting corresponding features within our hand. The location of the area representing our finger can be determined by a precise calculation of the orientation of our hand. The size of the area is evaluated by defining geometrical dimension markers, simulating a virtual ruler in three dimensional space.

The fundamental steps in order to separate the fingers from the hand are illustrated in 38. We start by determining a centric region along the length of the hand's mask (a). This procedure is used in order to realize the definition of the area, that our dimension mark-

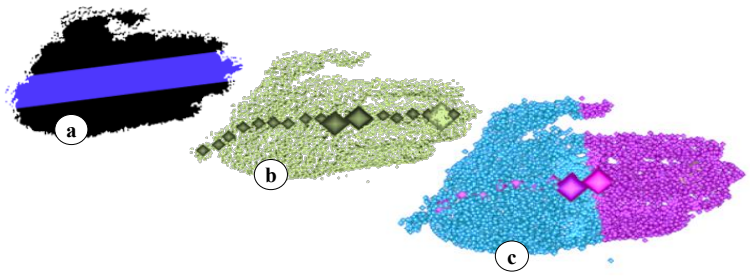

Figure 38: Overview of feature detection: our solution starts with a centric aligned region within our hand (a). We define dimension markers (b) simulating a virtual ruler, in order to separate the fingers from the hand (c).

ers are supposed to lay in. The dimension markers define "anchors" within our "wobbly" point-cloud, identifying fixed points. We implemented the detection of 20 markers (b), which are homogeneously distributed and centrally arranged. By the use of the dimension markers, we are able to define a partition wall and isolate the region within our mask representing the fingers (c).

The Centric Aligned Region

We need to correct the hand's orientation which is calculated in prior components, in order to determine a centric aligned region. We defined the desired corrected orientation as a vector spanned from the spatial moment to the position of the longest finger in two dimensional space. Hence we need to investigate the area the fingers are supposed to lay in. We reuse the concept of the virtual viewport in order to determine the current region of interest. The corresponding implementation is illustrated in Figure 39. By iterating through the pixels within this region of interest, we are able to determine the location (a), which is the farthermost away from the spatial moment (b). The vector stretched from the spatial moment to the detected position of the fingertip, represents our corrected orientation (c) in two-dimensional space .

In order to define the desired centric aligned region we implemented an additional virtual viewport as illustrated in Figure 40. This viewport is mounted to the longest fingertip and points into the opposite direction of the corrected orientation. Our investigations

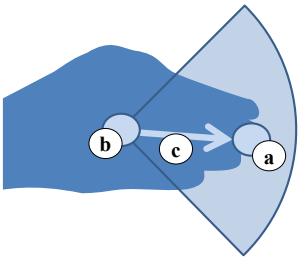

Figure 39: The corrected orientation: is represented by a Vector (c), spanned from the spatial moment (b) to the longest fingertip (a). This Fingertip has to be within the virtual field of view and the farthermost away (a) from the spatial moment.

found a field of view of 15 degree as convenient in order to cover an area the dimension markers are supposed to be placed.

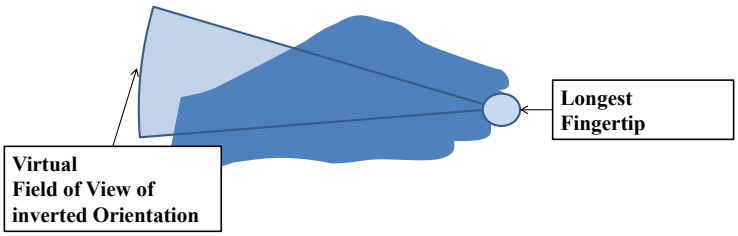

Figure 40: The centric aligned region

Identification of Dimension Markers

The dimension markers operate as a twenty cm ruler, which is centrally aligned along our hand lengthwise. In order to locate those markers and calculate their position within our physical environment, we need to make use of the three-dimensional data of our mask. The starting point marker is detected as the fingertip that is the farthermost away from the spatial moment within our two dimensional mask. We calculate the three-dimensional local mean of this fingertip in order to mount our ruler to the hand. The dimension markers need to be placed roughly equidistantly to each other, in order to simulate a virtual ruler. We found 1 cm as an appropriate unit in length, each marker has to be in distance to its predecessor. Hence each marker's index reaching from 0 to 20, indicates its three-

dimensional distance to the fingertip, as a multiple of 1 cm. In order to detect the position of each marker, we iterate through the pixels which are within the centric aligned region and analyse the corresponding three dimensional data. The resulting feature construct is illustrated in Figure 41.

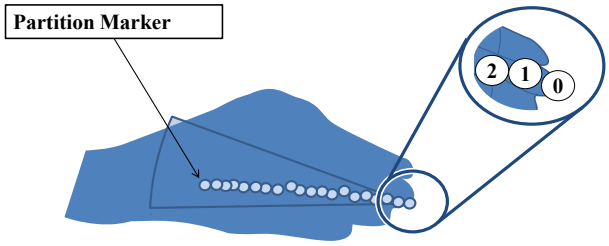

Figure 41: Dimension markers placed within the centric aligned region: Based upon three dimensional coordinates, we fixed each marker of our virtual ruler within the centric aligned Region

Isolating the Fingers

The detection of dimension markers enables the isolation of the set of pixels representing the user's fingers as illustrated in Figure 42. We use a two-dimensional vector as a partition wall, separating the relevant set of pixels. This vector is orthogonal in reference to our corrected orientation and need to be fixed to one of the detected dimension markers in two dimensional space. The marker that is appropriate as a position vector, is determined by the length of the users finger. The average length of the user's finger has been detected within the component which monitors the initialization gesture, described in Section 5.1.2. This information is reused in the current step of our procedure. Rounding up the length of the finger to integer identifies the corresponding index of the desired dimension marker. If the average length of the finger is between 6 and 7 centimetre, the partition wall is fixed to the 7th dimension marker as illustrated.

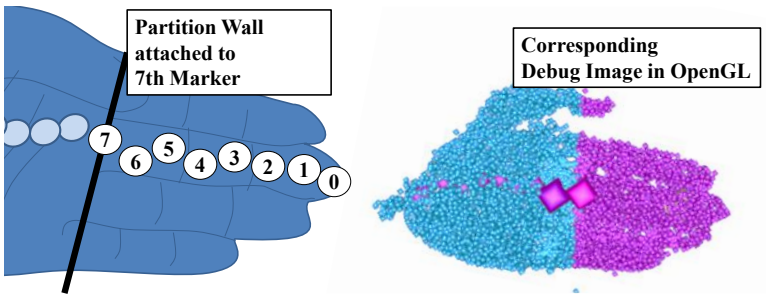

Figure 42: Isolating the finger: The Finger are separated from the palm by a partition wall attached to the 7th dimension marker.

5.3.3 *Analysing the hand's surfaces*

The following subsection presents our solution to identify the three dimensional orientation of our hand as illustrated in Figure 43

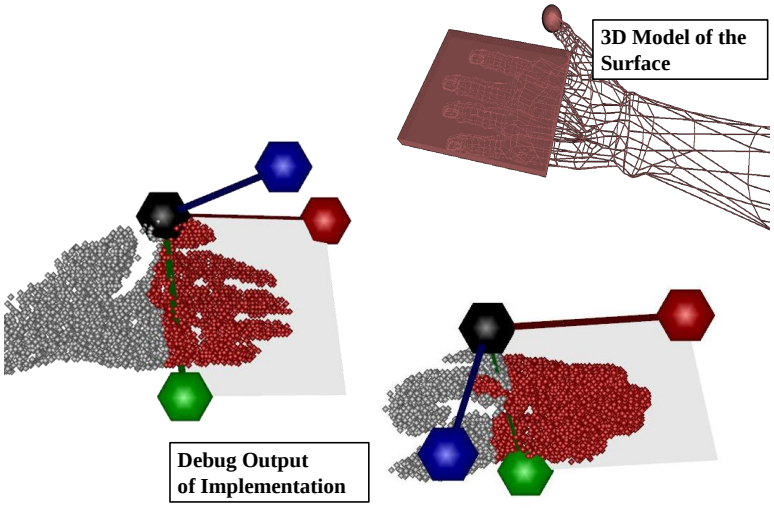

Figure 43: The finger as an input surface: We implemented a plane detection algorithm analysing the hand's posture.

The surface detection algorithm is based upon the point cloud representing the user's hand. This point cloud strongly deviates within three dimensional space. In order to span the desired plane in three dimensional space, our algorithm needs to identify the two vectors representing the direction the point-cloud deviates the strongest. We propose principal component analysis, calculating three eigenvalues of three eigenvectors, in order to approximate the hands plane within the point-cloud. Eigenvectors describe the square sum of the point's deviations along the corresponding axis. Hence the eigenvector containing the smallest eigenvalue defines the best-fitted plane's normal.

Since our point-cloud doesn't only deviate along the width and length of our hand, but also because of noise, skin folds and the shape of our finger, we need to investigate the set of pixels that are useful for the PCA-algorithm. Of course we use the region representing our finger, since it defines the user's input space. Further

we need to investigate additional areas that could approximately add to the stability of the resulting approximation. We propose the incorporation of the data representing a subset of the palm, since it contains equivalent orientation compared to the finger, as long as the user doesn't bend his hand in a significant manner. The thumb is supposed to represent an outlier in relation to our desired plane. Especially the muscles at the origin of the thumb lift themselves up, as the thumb operates upon the fingers. However we cant isolate the thumb completely yet, since we didn't detected its region by then. Compared with the hand, the wrist is roughly tube shaped and don't necessary represent the finger's orientation. Hence it is not advisable to incorporate the wrist's pixel or the rooting muscles of the thumb in order to approximate the hand's plane. Our investigations found the sum of pixels representing the fingers and roughly the half of the palm relevant for PCA. Therefore we implemented a partition wall within the mask of our hand, in order to isolate the interesting pixel from the useless. This vector needs to be orthogonal to the orientation and fixed to a specific dimension marker. Our solution is illustrated in Figure 44.

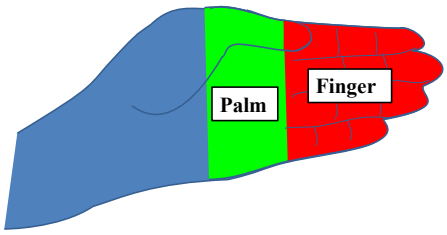

Figure 44: The reduced regions of interest for PCA: The finger (a) and a fragment of the palm (b) is applicable for PCA and isolated from the unneeded regions.

We propose to multiply the average finger length by 1.5, to detect the most suitable marker the partition wall is fixed to. The mean of the relevant pixels in three dimensional space, is used in order to define the position vector the plane is mounted to.

5.3.4 *Detecting the Thumb*

The following section describes our solution to separate the thumb from the hand. Figure 45 illustrate the relation between the two components representing the thumb (red pixel) and the hand (blue pixel). The thumb is successfully isolated within the image, even if the is held in different postures.

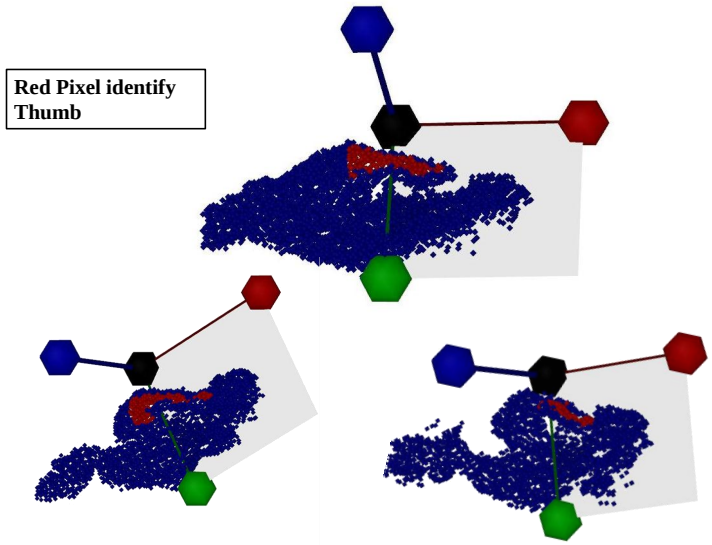

Figure 45: The thumb isolated from the hand: Red pixels represent the thumb within a blue point-cloud illustrating the hand.

The subset of pixels representing the thumb and the subset of pixels representing our inputspace merged together to a global set, that was used in order to approximate the surface of the finger. As the component of the users thumb demand way less pixel then the component of the inputspace, we can assume the plane to be more sensitive to the orientation of our finger. Hence the thumb's position has a relatively small effect on the result of the PCA and becomes a statistical outlier in reference to the approximated plane.

The Figure 46 illustrates the relation between the posture of the hand's surface and individual pixels in three dimensional space. We calculated the standard deviation of the distance of all pixels in relation to the plane. Grey coloured pixel represent measurements within standard deviation to the plane. The color red refers to indi-

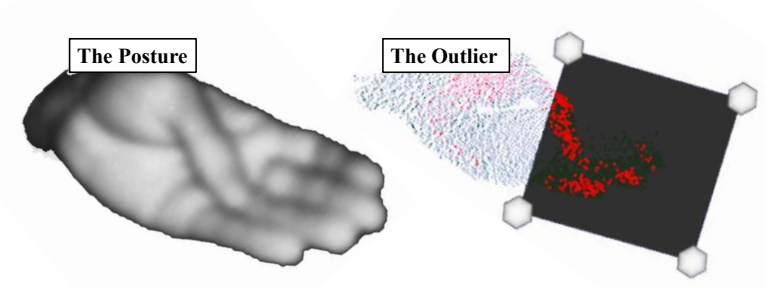

Figure 46: Detecting the pixel representing the thumb: The grayscale image displays the posture of our hand. Each pixel's color within the corresponding point-cloud refer to its relative distance to the approximated plane. Red coloured pixels indicate that the individual measurement is above standard deviation in distance to the plane.

vidual pixel, which are farther than the standard deviation. Adjacent areas of red coloured pixels, indicate potential regions representing the thumb. However, there are multiple regions of pixels that are beyond standard deviation in distance to the plane. Hence it is crucial to separate the component that represents our thumb from adjoining areas containing similar effects.

In order to separate the region that represents the thumb, we need to check each regions logical validity based on prior assumptions. Therefore we map the image, containing the data of the distance of each pixel in respect to the plane, to a two-dimensional image as illustrated in Figure 47. This mapping enables the possibility to reuse OpenCV's functionalities, in order to analyse the features of connected areas of interest.

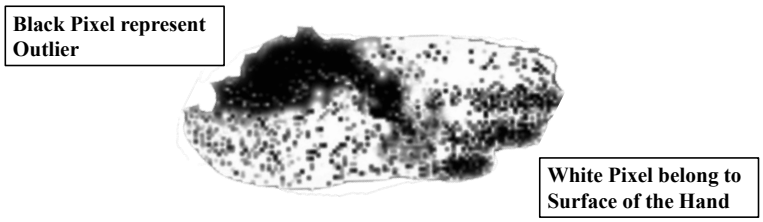

Figure 47: The *Distance to Plane Image*: Each pixel's greyscale value represent its distance to the approximated hand's surface. Dark pixel refer to a relatively high distance and indicate potential regions representing the thumb.

Our investigations found the following feature definition, in order to detect the thumb. The region of our thumb is rarely interrupter by significant skin folds and reaches from the hand above the inputspace. Therefore the component representing the thumb has to obtain the largest area, of all regions which overlap the region of the inputspace and the rest of the hand. The corresponding procedure is illustrated in Figure 48. The image (a) is a model of all potential regions of interest (red). We reused the concept of the partition wall (b), described in section sub:Detection of hand and thumb interaction:Detecting the features of the hand in order to fragment the hand from the inputspace. The thumb (c) is detected as the biggest area which lays within both fragments.

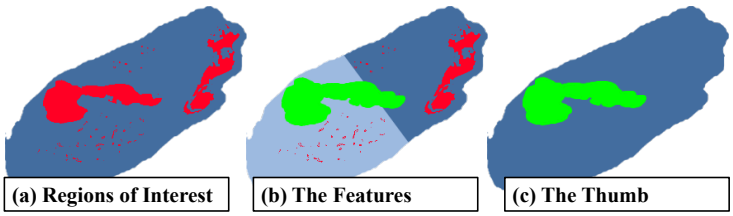

Figure 48: The regions of interest: The hand mask (blue) in reference to potential areas representing the thumb (green).

The following Figure 49 illustrate the greyscale image of different hand postures and their corresponding point-clouds. Within the point-clouds, blue pixels represent the area we detected as the thumb. The images (a) to (c) display the hand held at different angles along the roll axis. The image (d) displays the hand as the fingers are crunched into the direction of the camera. Even under this marginal condition our algorithm is able to detect the region representing the thumb.

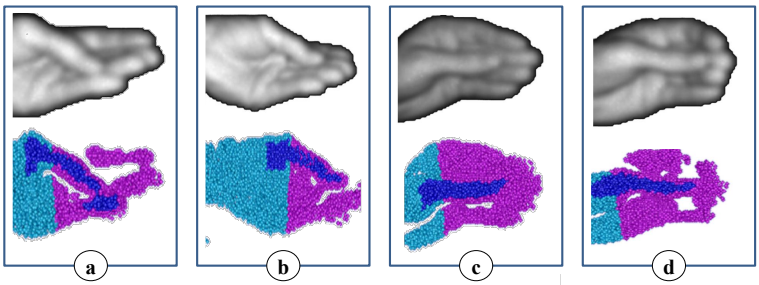

Figure 49: The thumb detection: The greyscale images illustrate different hand poses. The corresponding point-cloud below represent their visualization in OpenGl. The blue colored pixels illustrate the detected thumb region.

Detecting the Tip of the Thumb

The following Section presents our algorithms in order to detect the tip of the thumb. The debug output images in Figure 50 illustrate the result of our detection.

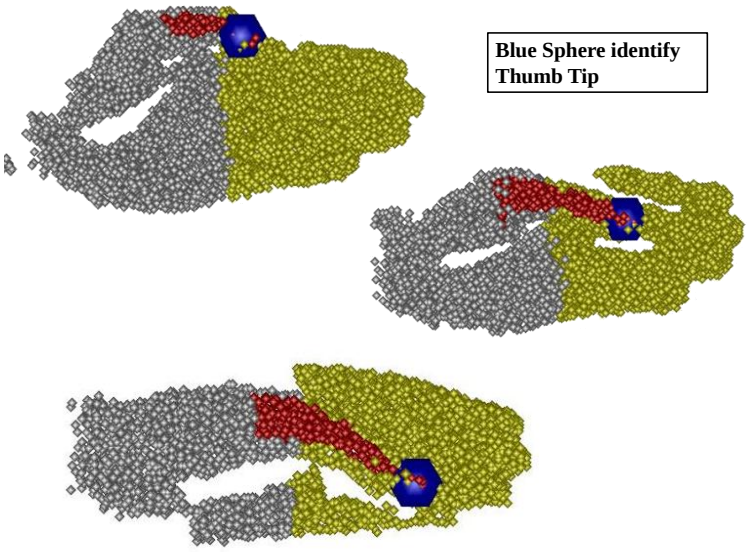

Figure 50: Detecting the thumb tip

In order to detect the tip of the thumb we need to proceed by creating a dynamic skeleton of our thumb. This skeleton correlates to the topological features of our component representing the thumb The main topological features of this component are listed below and have to be identified in the corresponding order.

- The root of the thumb (a). This point corresponds to the actual location where the thumb roots out of the hand.
- The middle point of the thumb (b), roughly located around its knuckle.
- The tip of the thumb (c).

Our solution in order to detect the tip of the thumb is illustrated in Figure 51. The point (a) is the farthermost away from the longest fingertip (b) and identified as the root of the thumb. We described our solution in order to evaluate the longest fingertip in Section

5.3.2. We continue by calculating the spatial moment (c) of the thumb component. This feature will be identified as the middle point of the thumb. We use the detected root point and the middle point in order to span a vector between the two. This vector is used, so we can fix a virtual field of view (d) to the spatial moment of the thumb, identifying the region where thumb tip is expected to be. Iterating through all the pixels that represent the thumb and are within the virtual viewport, enable the possibility to detect the pixel which is the farthermost in respect to the knuckle point. This position represents the tip of the users thumb (e).

Figure 51: The procedure of thumb tip detection

5.3.5 *The thumb in reference to the hand*

The following subsections present our procedures in order to enable a touch pad analogy, as the user's thumb navigates across the hand. We detect the thumb tip's position in relation to the hand's surface as illustrated in the debug output collage of Figure 52.

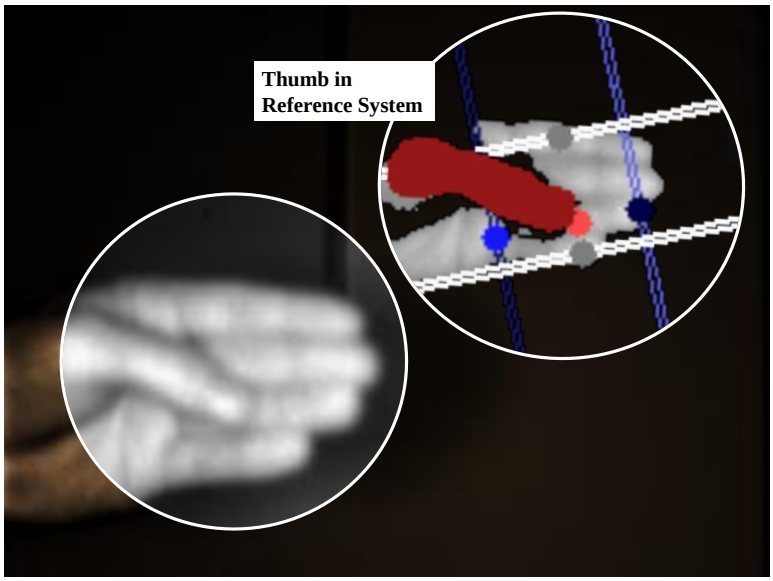

Figure 52: The model of the hand. A plane is approximated within the Hand (green). Three geometrical axis are fixed to the origin of a coordinatesystem

Our solution was initially intended to be based exclusively on the three-dimensional features of our hand. The orientation of the plane of our hand's surface, the origin of our reference system and the three-dimensional position of our thumb tip must be used in order to create the desired touchpad analogy in three-dimensional space. Unfortunately this approach is limited due to the capability of contemporary depth cameras. Measurement noise and low resolution are accountable for the restrictions of our initial concept and motivated us to investigate alternative approaches. Hence we propose an interplay of two dimensional and three dimensional data in order to detect the interaction between thumb and hand. Our research found the following solution of feature detection as the most promising:

- The 2D Position: The position of the tip of the thumb in relation to the hand is determined by the use of the two dimensional mask of the hand as illustrated in Figure 53. Since we expect the user to hold the hand roughly orthogonal to the chest, we are able to determine the position and dimension of the hand's surface. This enables the evaluation of the thumb's relative position to the hand.

- The 3D Tap: We determine the Hesse normal form of the hand's plane, in order to calculate the distance of the thumb tip in reference to the plane. If the resulting value is less then 1 cm, our system registers the thumb to be in contact with the hand.

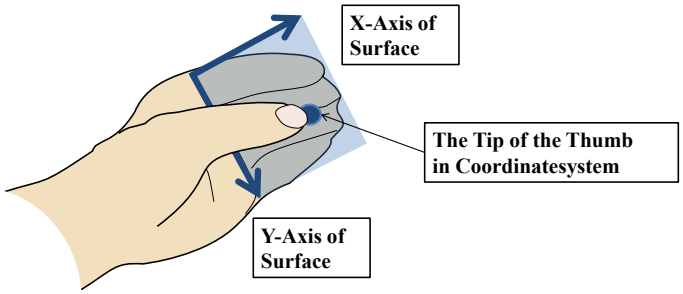

Figure 53: The two dimensional reference system.

Our model illustrated in Figure 53, led quickly to promising and smooth results. The only limitation arise by the trapezoid shape of the user's hand. The finger's inputspace can't be mapped to a rectangle, especially since it strongly differ from user to user. Hence it is necessary to implement a generic configuration of the inputspace, which adapts to the user's hand's shape.

Our solution in order to reconfigure the inputspace is illustrated in Figure 54. The origin of the desired relation system is defined by the dimension marker, which corresponds to the finger's average length. We described the relevant procedures in section 5.3.2. We use the concept of virtual field of views, in order to continuously reconfigure the dimensions of the user's inputspace. The maximum of the local reachable space along the x-axis is evaluated by fixing a virtual viewport to the thumb tip (a). This viewport points into the direction

of the hand's orientation and covers the corresponding pixels at the contour of the hand. We use all of those pixels in order to calculate their two-dimensional mean (b). The distance between the mean and the y-axis (c) identifies the maximum of local reachable space (d). The size of that space in reference to the distance (e), between the mean and the tip of the thumb, identifies the relative position of the thumb tip.

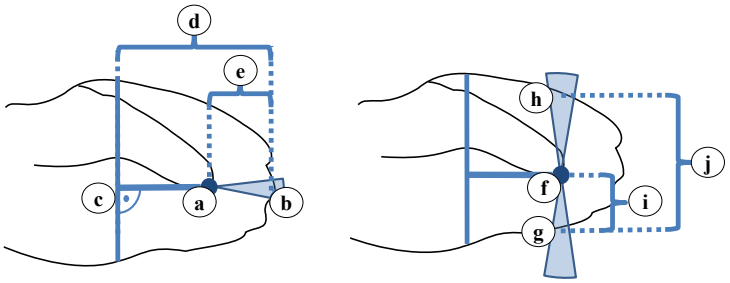

Figure 54: Recalibration of the axis. The tip of the thumb (a) in reference to the range (d) between the fingertips (b) and the y-axis (c). The distance (e) between the fingertip and the tip of the thumb in respect to the range defines the relative position of the Thumb.

Our solution, in order to recalibrate the length of the y-axis, is analogue to the calibration along the x-axis. We use two viewports, mounted to the thumb tip (f). We calculate the corresponding means (h and g). The distance between the two define the local maximum reachable space (j) along the y-axis. The distance (i) between the tip of the thumb and the lower mean (g) in relation to the local maximum, represents the thumb's relative position. We used the lower mean, because its contour is naturally less frequently manipulated by the thumb's pixels.

The Figure 55 displays our successful recalibration of a two-dimensional coordinate system as the user's thumb navigates across the hand's surface.

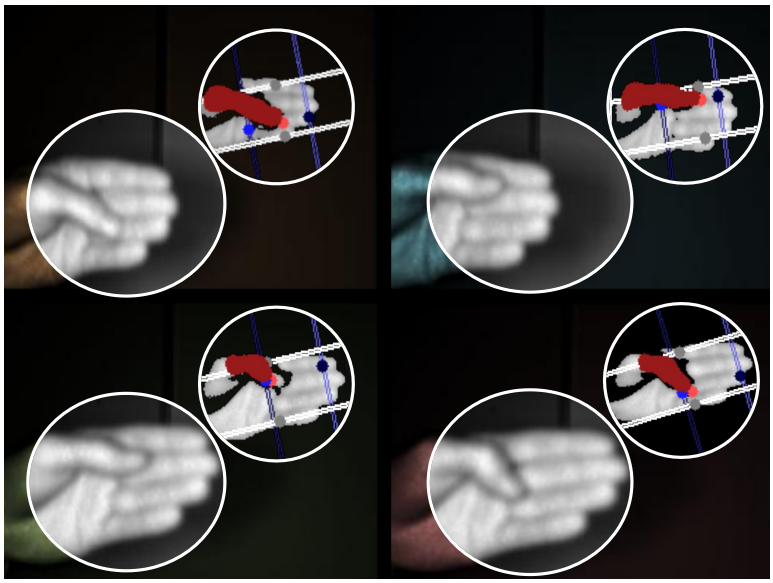

Figure 55: Result of recalibration of a two-dimensional coordinate system displayed by debugging-pictures and their corresponding hand-poses

5.3.6 *The thumb in reference to individual fingers*

Our investigations found effects, which offer the possibility to detect the thumb tapping individual fingertips as illustrated in Figure 56.

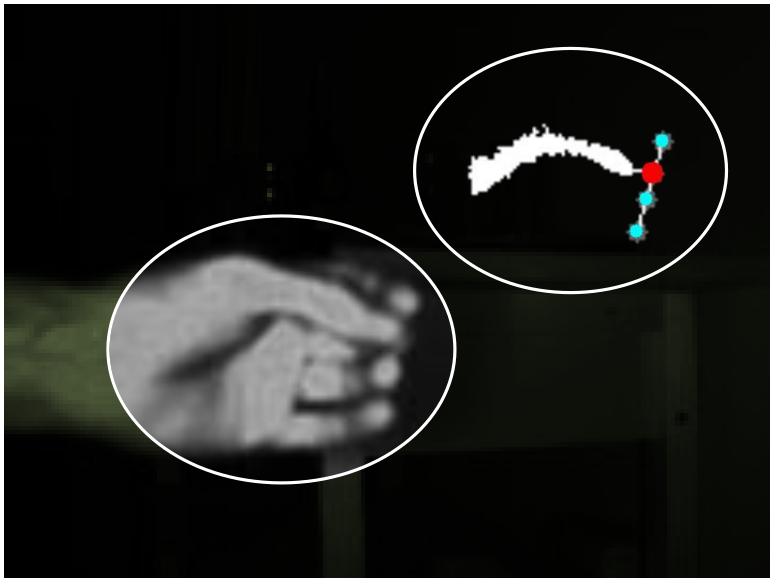

Figure 56: The Thumb tapping a Fingertip: The grayscale image of our hand illustrates the thumb tip tapping the middle finger. The corresponding debug output image displays the second item from the top to be selected (red), which refers to the middle finger.

If the user holds the fingers slightly bent towards the camera as illustrated in Figure 57 (a), then shadows appear around the bended finger joints. Those shadows are caused by folds of the skin, which don't reflect enough infrared light, and separate the finger from the hand within our image. Figure 57 (a) illustrated the fingertips, which are clearly isolated from the rest of the hand. Unfortunately the finger's pixels connect and disconnect from each other, depending on the hand's pose in reference to the camera. Because of that, it is not possible to identify individual fingertips and their position within our image. Instead we use this mask in order to determine the sum of pixels representing the fingertips as one single component and reconstruct the position of each tip within this region.

In order to approximate each fingertip's position we need to detect the features illustrated in Figure 58. We define the components

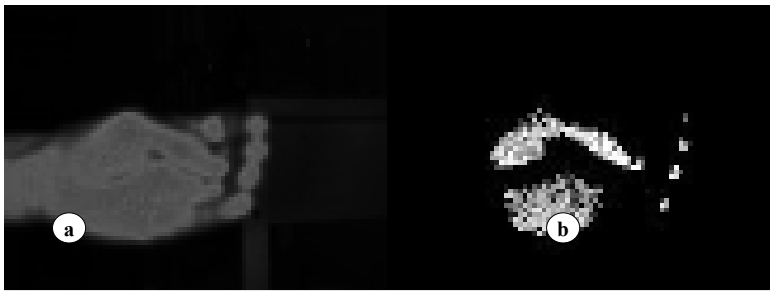

Figure 57: The Effect as the hand is bend: Shadows clearly isolate the fingertips from the hand as the finger are bend towards the camera.

representing the fingertips as one single component (a). The spatial moment and the orthogonal of the hand's orientation is used in order to span a vector (b), representing the approximated row the fingertips are aligned to. This vector enables the possibility to measure the width spanned by the users fingertips (c). By segmenting the area in four regions, we are able to approximate the four fingertips along the vector.

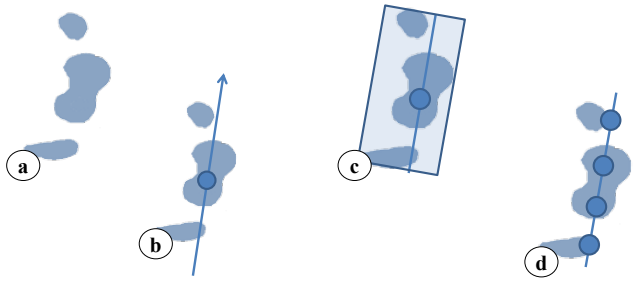

Figure 58: Reconstruction of the Fingertips: We identify the pixels representing the fingertips as a single component (a) and calculate its spatial moment (b). The orthogonal of the hand's orientation fixed to the spatial moment offers the possibility to measure the width of the fingertips and identify the approximated position of each finger.

The Figure 58 illustrate our solution to detect the thumb tip's relative position in relation to the fingertips. We reconstruct the direction of the thumb's orientation by the use of the vector, which is spanned from the spatial moment of the thumb to the tip of the thumb. This vector intersects with the vector representing the row of the aligned fingertips. The Fingertip which is the closest to this intersection represents the item the user desires to tap. The

discrete tap is registered as soon as the thumb tip's distance to the intersection point is below a lower boundary.

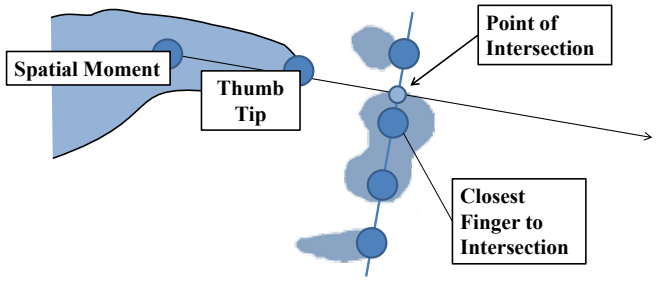

Figure 59: Features of thumb and fingertips: the closest fingertip to the thumb is registered.

6

APPLICATIONS CONTROLLED BY IMAGINARY WIDGETS

The following chapter presents our solutions, to translate the detected features of the gesturing hand into imaginary widgets. The resulting continuous and discrete values control the application's input as illustrated in Figure 60. We implemented audio and graphic applications in order to demonstrate multiple fields our system could become advantageous. Our applications are written in Java based Processing. We describe the imaginary widgets which were built on top of the pinching gesture in section 6.1. The widgets of thumb on hand interaction are illustrated in section 6.2.

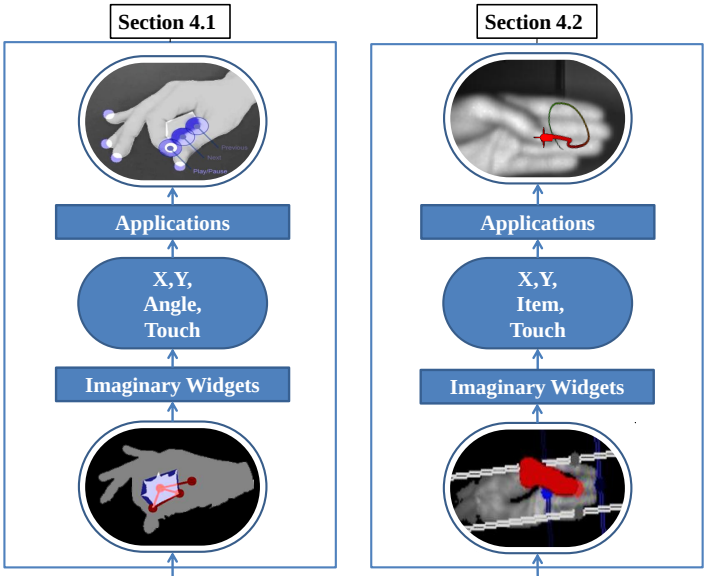

Figure 60: Overview of application and widget architecture

6.1 IMAGINARY WIDGETS BASED ON PINCH GESTURES

The following section presents our implementation of imaginary widgets based on pinching gestures. The feature detection of the pinch, which is presented in section 5.2.2, provide the essential data to construct the widgets. We implemented an audio player in order to investigate the complexity of our input technology. This application offers the possibility to switch between different songs by discrete input and adjust the volume by continuous input. Hence our pinching widget realizes two different types of imaginary widgets. The imaginary dial returns its rotational angle. This continuous value determines the level of the volume. The imaginary item bar returns the index of an option that is currently selected and controls the playlist. Additionally the widget component verifies if the user performed a double-tap. If this condition becomes true the user is enabled to control the playlist. If the user performs a single-tap, the continuous data of the dial is parsed to the Volume control of the audioplayer.

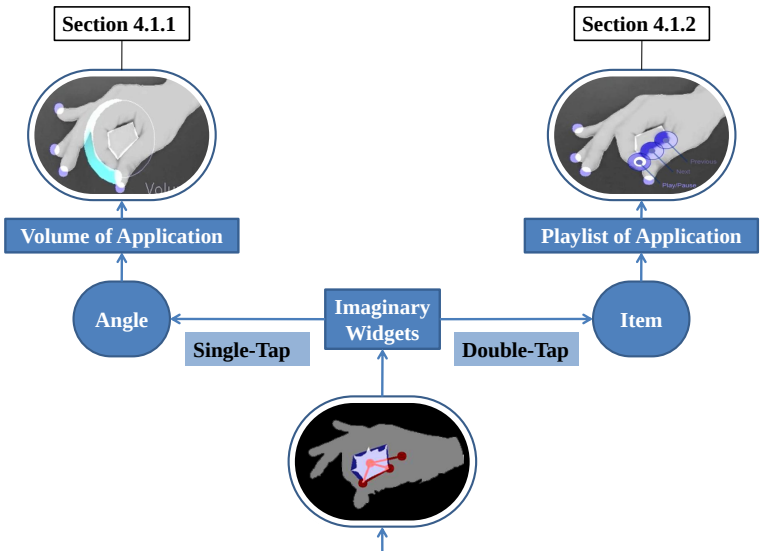

Figure 61: The architecture of the imaginary pinching widget: Continuous and discrete values are identified and forwarded to the corresponding control assembly of the audioplayer. Which assembly is supplied with data, depends on whether the user performing a double-tap or not.

The check procedure of the double-tap is realized, by simply checking the state of the pinch over time. Discrete data is provided by the procedures described in section 6.1.2. The functions in order to determine the continuous input values are discussed in the following section 6.1.2.

6.1.1 *Continuous Input*

Our solution in order to detect the continuous values of the pinching widget is illustrated by the debug output collage of Figure 62. The volume is adjustable by an imaginary dial which turns as the user's index finger slides along the thumb.

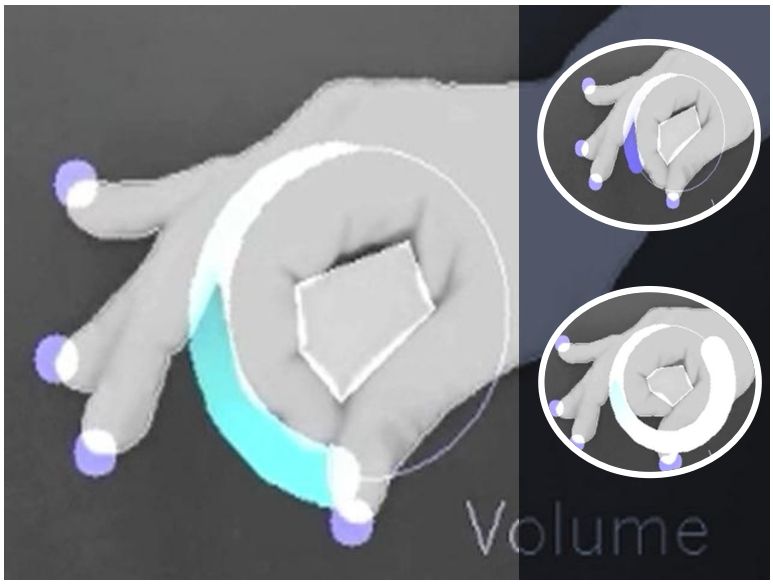

Figure 62: Controlling the volume of the audioplayer: The level of the volume is adjusted by sliding the tip of the index finger along the thumb

The corresponding calculation in order to detect this continuous value is illustrated in Figure 63.

- We span a vector (a) from the spatial moment of the component of the hand to the spatial moment of the pinch component.
- We span a second vector (b) from the spatial moment of the pinch component (c) to the point of contact between thumb and index finger.

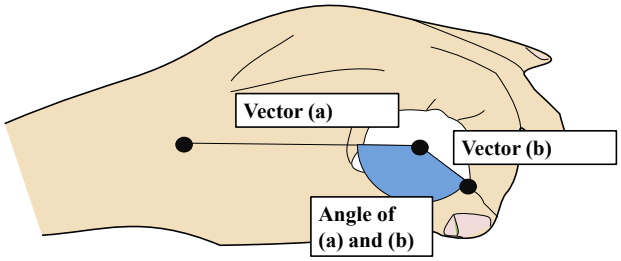

Figure 63: The calculation of the imaginary dial: The angle between the two vector's (a and b) define the dial's rotation

- We calculate the angle between the two vectors (a and b).

We use the value of the resulting angle in order to control the volume. Our solution offers multiple advantages. First of all, the spatial moments are very robust to noisy measurements and fragments appearing within our image matrix. The more our algorithm is based upon robust features, the more precise is the result of our solution. Another even more interesting advantage of this concept is illustrated in Figure 64. A dial between the user's thumb and index finger, can be rotated in one fluid motion as the index finger slides along the thumb (a). As the index finger reaches the tip of the thumb (b), the user can continue to slide with the thumb tip along the index finger (c). This motion still causes the rotational angle to grow and increases the desired value. This advantage expands the range of the inputspace, in contrast to only using the thumb.

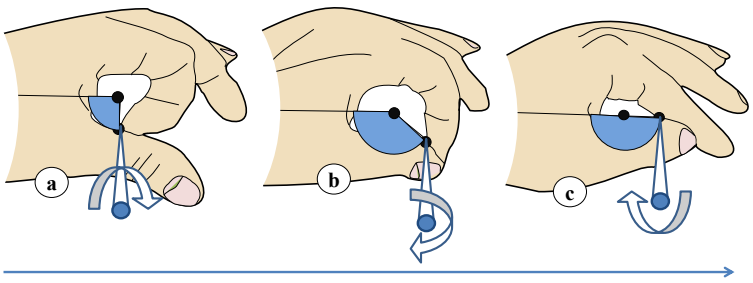

Figure 64: The imaginary dial's rotation: The rotation is continuous in one direction as the user performs the illustrated sequence of gestures from left to right.

6.1.2 *Discrete Input*

This subsection illustrates our solution to extract discrete input, as the user performs the pinching gesture. Figure 65 presents a collage of debug images of our imaginary item bar implementation. This widget is called by double tapping and holding the tip of the index finger on top of the thumb. The user now continues to slide his index finger along the thumb and imagines different items to be placed on top of it. In order to select one, the user simply releases the contact between thumb and index finger close to the position he imagines it to be placed. The active item that was the closest to the release point of the two fingers, will call the corresponding functions within the application.

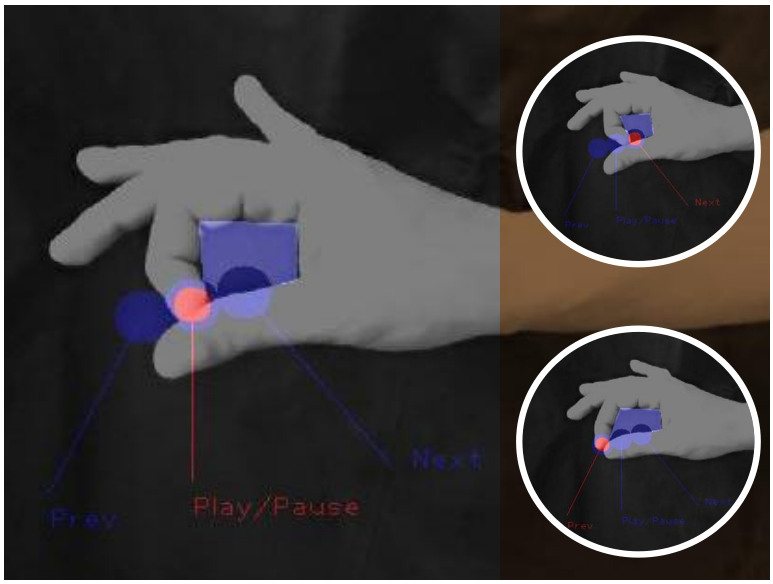

Figure 65: Playlist of audioplayer controlled by imaginary items: An item bar is fixed to the thumb, representing different options in order to play or skip between different songs.

Our concept to determine the position of the individual items is illustrated in Figure 66. As soon as the user double taps, we calculate the vector and distance between the point of pinch and the point where the thumb roots out of the hand (a).

We proceed by attaching different items the user can select from (b). Since the resolution of our camera and the available input space on top of our thumb is relatively small, we decided to stick to a

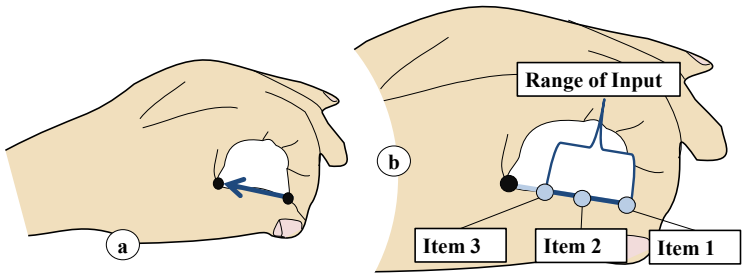

Figure 66: The imaginary dial's rotation: The rotation is continuous in one direction as the user performs the illustrated sequence of gestures from left to right.

minimum of imaginary items to be placed along our thumb. One of those items needs to control the pause and play button. Two more items need to offer the possibility to select the next or previous track within our play list. Therefore we fragment the inputspace on top of the thumb down to three regions, representing three imaginary items. Unfortunately it is not possible to use the hole inputspace on top of the thumb, since the user could possibly close the pinching hole as soon as he tries to reach the root of the thumb with his index finger. Hence we propose to only use 75 percent of the inputspace. We perform this kind of calibration, which is responsible for the distribution of the item's position , only once the double tap is performed. This initial calculation and its results is necessary in order to measure the available input space on top of our thumb. Therefore it is recommended that the user always double taps the thumb as close to its tip as possible. This stretches the available input space to a maximum, thus increases the quality of fragmentation. Once the user revealed the length of his thumb by his initial double tap, the discrete items are fixed on top of his thumb in reference to the static position where the thumb roots out of the hand.

6.2 IMAGINARY WIDGETS BASED ON THUMB AND HAND INTERACTION

This section presents our solution, concerning the translation of thumb and hand interaction into corresponding imaginary widgets. The following Figure 67 illustrates a debug output collage of the user's hand interacting with a visual application

Figure 67: Thumb on hand widget: The user pushes different imaginary buttons on his hand, which cause a little hairy ball to change his haircut.

We identify the individual Gestures by the use of the features, we detected in Section 5.3. We are able to determine:

- Continuous values as the thumb navigates across the fingers. We detect its position based on a two dimensional coordinate system mapped to our hand mask.
- Discrete values as the thumb taps the fingers. We continuously calculate the distance of the thumb in reference to the hand. Since the stability of this value suffers from noisy depth measurements, we need to smooth our data. An exponentially weighted moving average across the latest 7 frames offers the desired reliability and minimizes the latency within an appropriate amount. Our system registers a touch event, as soon as

the result of the smoothing algorithm calculates an average below 1 cm.

- Discrete values as the user taps individual Fingers.

The following subsections describe the implementation of individual imaginary widgets and their corresponding applications. We created graphical illustrations, in order to demonstrate the efficiency of our interactions technologies visually.

6.2.1 *Continuous Values*

This subsection illustrates the efficiency of our interaction technology concerning continuous thumb on hand interaction. The continuous values of the imaginary thumb on hand widgets are used in order to control different floating objects on the screen.

The Slider

We implemented different imaginary sliders, mapping the data of the feature detection to visual applications. Figure 68 illustrates the famous video game Pong, which is controlled by the user's thumb tip sliding across the finger. Hence a small paddle, which is supposed to hit a white ball, corresponds to the y-coordinate of the thumb tip in reference to the user's hand. The images illustrate different positions of the thumb and the paddle, which is associated to it.

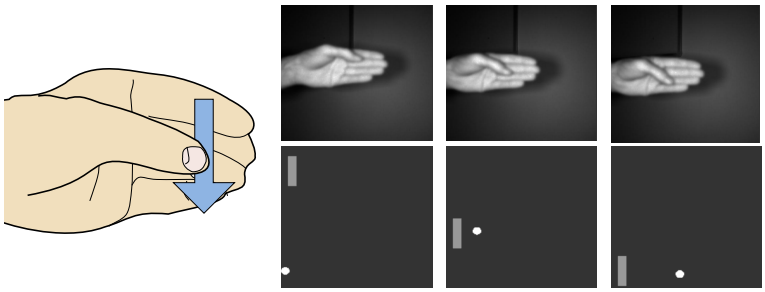

Figure 68: Pong controlled by imaginary slider: A small Paddle is controlled by the user's thumb tip, moving across the hand.

We designed an additional slider, in order to demonstrate the ability to interact with the thumb as it moves along the finger. Our solution is illustrated in Figure 69. The game consists of a red ball, that bounces on top of a virtual street. The game is lost if the ball falls into a black hole. The horizontal position of the ball refers to the x-coordinate of the thumb in reference to the hand.

The Touchpad

We designed a motion flow visualization in order to demonstrate the imaginary touchpad. The position of our thumb in reference to the hand is related to the controlled pointer within the drawing frame.

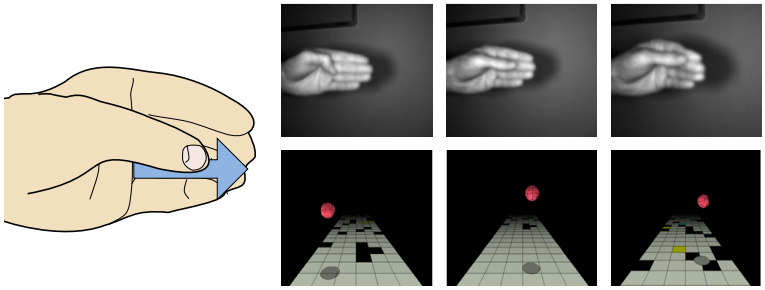

Figure 69: Ball game controlled by imaginary slider: A red ball corresponds to the thumb tip sliding along the finger.

This concept is illustrated in Figure 70. A continuous line illustrates the temporary motion of the thumb as it moves across the finger's surface. A red cross is placed at the actual position, indicating the contact between the thumb tip and the finger.

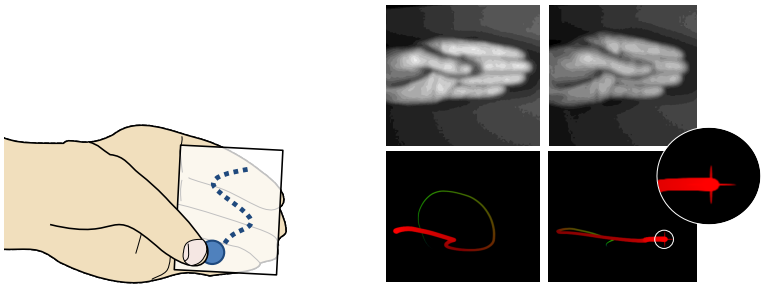

Figure 70: Motion flow across imaginary touchpad: The line illustrates the movement of the thumb tip. A red cross indicate the touch event.

6.2.2 *Discret Values*

This subsection illustrates the efficiency of our interaction technology concerning discrete thumb on hand interaction. The discrete values of the imaginary thumb on hand widgets are used in order to select objects on the screen.

The Dial

The imaginary discrete dial illustrated in Figure 71 , enable the possibility to select from different items, which are rotationally organized. The user iterates through those items by performing a circular motion with his thumb tip across the finger's surface. A circular sector which is coloured in red, illustrate a touched item. A turquoise arc represents an item the user's thumb tip hovers over. This imaginary Widget is also implemented as a continuous widget. A white triangle is illustrated within the debug images, pointing at the relative position of the thumb tip. The rotation of this object, represents its corresponding continuous value.

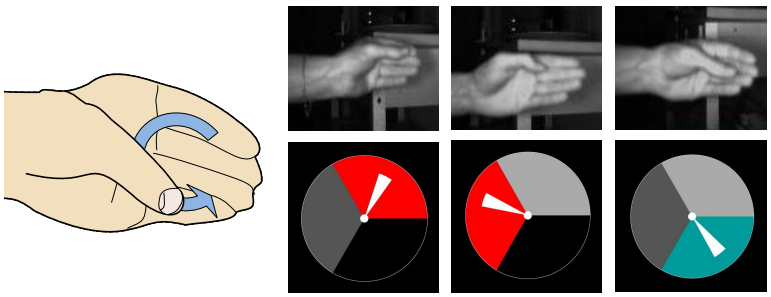

Figure 71: Discrete dial controlled by thumb on hand interaction: The user's thumb tip performs a circular motion on top of his hand. Red arcs represent touched items and turquoise arcs illustrated the hovered ones.

The Buttons

We implemented imaginary buttons the user imagines to be placed across his fingers as illustrated in Figure 72. Hence we fragmented the available inputspace in eight squares of equal size. Because of low resolution and the corresponding inaccuracy we decided to

avoid the implementation of more buttons for our prototype. Red coloured buttons represent a selected item as the user touches the corresponding area on top of his hand. A purple coloured button illustrates a hovered item.

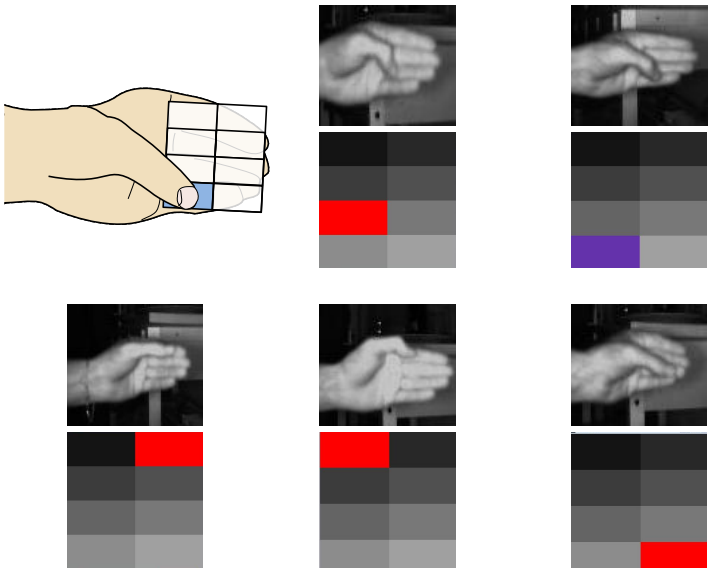

Figure 72: Discrete buttons controlled by thumb on hand interaction: The user is able to select (red) from different items by the use of buttons he imagines to be placed across his hand. As long as the user's thumb hovers across the thumb, the corresponding item (purple) is registered as not selected.

The Fingertips

We implemented imaginary buttons, which are placed on top of the users finger. The features detected in 5.3.6 determine which finger is actually tapped. A visual game in which different discrete signals determine the haircut of a hairy ball and its interactions is illustrated in Figure 73. Each haircut is assigned to an individual finger.

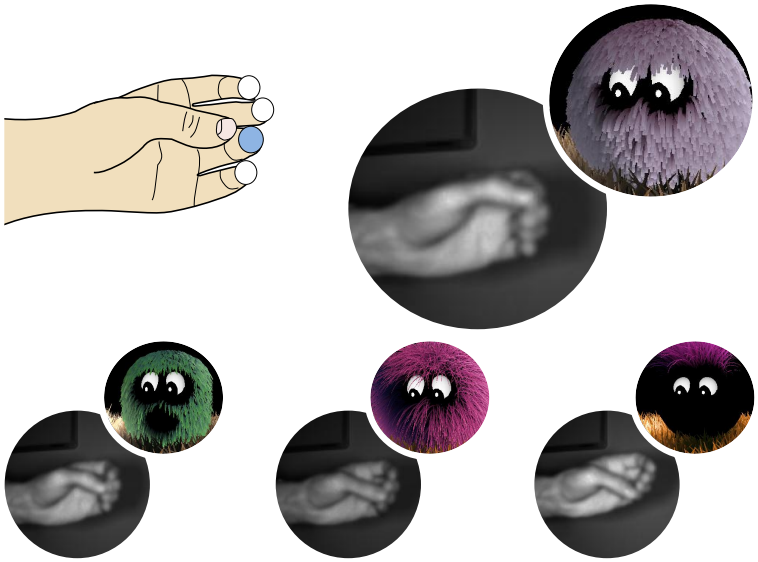

Figure 73: Haircut of hairy ball controlled by fingertips: By tapping an individual fingertip the corresponding haircut appears.

7

DESIGN DECISIONS

This chapter discusses different design decisions, which we had to run through in order to develop our system. We explore the properties of gestural interaction techniques in Section 7.1 and suggest single handed gestures which offer tactile feedback as appropriate for quick interactions in mobile scenarios. Optical gesture detection technologies for mobile scenarios, allow the user to interact with empty hands and quickly return to their primary task at all time. The Section 7.2 presents our experience and research in different optical solutions for the detection of hand and thumb interaction. Depth Cameras offer encouraging results and possibilities in order to detect objects and gestures. However there are still some limitations. We illustrate the general course of implementation and the trade off between the physical restrictions of the hardware and the minimal demands of our interaction technologies in Section 7.3.

7.1 GESTURAL INTERACTION

The input concept presented in this work addresses different restrictions and interaction drawbacks of common mobile devices. Our goal is to reduce the time frame and the intensity of interaction, which mobile devices command. We propose gestures as appropriate for mobile scenarios in order to implement the desired reduction. The implementation of gestures, as an input technology, enable the possibility to leave the user's hand empty at all time. Hence the user is able to quickly access and launch the device, perform single purpose gestures and return to his primary task.

In order to create a gesture input technology it is necessary to define the gestures and functionalities, which are suitable for the individual needs of the target system. Optical gesture detection for mobile scenarios demand two major design decisions. We need to narrow down the subset of gestures that is appropriate in order to control individual widgets and we need to develop mechanisms in order to avoid unwanted activation. The following subsections discusses our design decisions to address both needs.

The Type of Gestures

Since our input technologies address different aspects of mobile scenarios, we need to identify a reasonable set of gestures which is appropriate for use in public. First of all we want to exclude bimanual interaction from further investigations, since we want to offer the user of our system to continuously free at least one hand from interaction. Single hand gestures enable the user to carry a coffee or hold a railing in the bus as they interact with the device with the other hand. We investigated two types of single hand gestures for our purpose. The group of "hand in respect to body gestures" and the group of "thumb in respect to hand gestures". Interactions which are defined by the hand's relative position/orientation to the body are often used for pointing and motion gestures. Those gestures are commonly detected by calculating the hand or forearm's position in respect to a

predefined reference point. In our case the corresponding detection system could be realized by the implementation of a body centred coordinate-system, which refers to the position of the gesturing hand. However there are multiple drawbacks for implementing such a gesture-set.

- The Reference System: A body centred reference system is not appropriate for mobile scenarios. The precise detection of the interacting object and the object which represents the origin of the relevant reference system is equally crucial. A moving body, carrying a camerasystem, is not necessarily synchronized with the sought-after motion of the hand's gesture.

- Discrete Data: The definition of continues input is often represented in computer vision solutions for gesture detection. On the other hand, extracting discrete data from a continuously moving hand, which is captured by a moving camera system is a crucial task. This condition is intensified by the fact that every gesture which is performed in order to initiate a discrete signal, might manipulate the corresponding continuous value. For example a touchscreen analogy in mid air is complicated, since every touch the user performs causes his hand to change its relative position. Therefore a false detection of the desired touching point is plausible.

- Social acceptability: Causing public attention and occupying a lot of interaction space with big movements of the hand and forearm, is not compellingly appropriate across all cultural barriers.

- Feedback: There is no sensing feedback, except the user watching his hand or feeling the muscles. A precise eyes free use of the gesture is very hard to perform.

We propose thumb and finger in respect to hand gestures. This concept is suitable for mobile scenarios as it positively addresses the drawbacks of the pointing and motion gestures:

- The Reference System: is determined by the hand itself. Hence the pointer, represented by the thumb, and its reference point can be determined within the same image frame.

- Discrete Data: The hand and finger represent a relatively static object in space, as the thumb operates on top of them. Hence discrete data is detectable by measuring if the thumb is in contact with the hand or an individual finger.

- Social acceptability: The imaginary input surface spanned across our finger remains a private area, because it is small and movable.

- Feedback: Humans are able to experience double tactile feedback as the thumb moves across the hand, since the thumb and hand deliver precise feedback about where the contact has taken place. Hence users are able to interact with imaginary widgets eyes free and perform precise input.

Avoidance of Inadvertent Activation

Random objects continuously appear within the chest mounted camera's view as we are on the go. Pedestrians passing by, objects of the surrounding infrastructure or the user's hands interacting with the environment. Those objects could possibly simulate the features of a hand and overcome any optical hand detection algorithm. Therefore it is necessary to develop a concept to avoid inadvertent activation. The following section describes the concept of an activation gesture the user's hand has to perform in order to proceed to further interaction. Such a Gesture needs to imply the following features.

- Security: The gesture should be unlikely to occur randomly. Different objects and artefacts of noise shouldn't be able to evoke optical effects, which simulate the activation gesture by accident. Therefore the gesture needs to reveal unique features of the hand in order to verify that it is a human hand. Additionally the device must not be activated by the user's everyday movements. Hence an appropriate gesture design and the corresponding

implementation needs to fulfil the following contradiction. On the one hand, the gesture needs to be as "naturally" as possible to the user and on the other side it needs to be quite unlikely to be performed in everyday situations.

- Usability: The activation gesture needs to be easy, comfortable and fast to perform.
- Social Acceptability: The gesture needs to be applicable across cultural barriers. This is especially needed since our concept is designed for mobile scenarios. Hence it is appropriate to develop a gesture that is small and must not provoke public attention or misinterpretations.
- Memorability: A sequence of movements which is used as an activation gesture, must not contain a complicated choreography. In order to push the memorability of our gesture, we need to develop a short redundant sequence of simple movements and reuse it for related functionalities. Therefore we propose the deactivation gesture to be the same as the activation gesture.

We propose a sequence of simple hand signs as illustrated in Figure 74 as adequate in order to fulfil the above mentioned demands for an activation gesture. Gestural interaction is unlocked once the conditions of each step of the sequence became fulfilled in the corresponding order and timing.

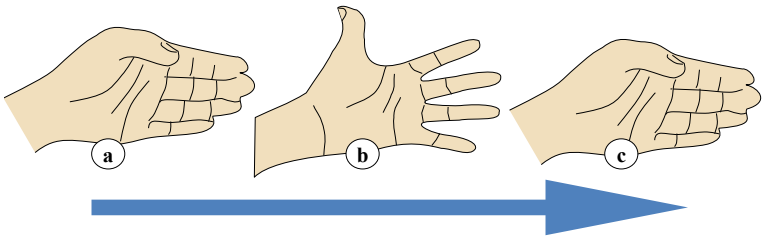

Figure 74: The sequence of gestures to initialize the system: a hand mask must be within our image(a). An open hand with spread fingers must be held for a specific timespan. Further the hand is expected to return into the position it was before (c).

7.2 OPTICAL DETECTION HARDWARE

In order to detect and analyse the gestures performed by the user, we investigated two different camera systems, capturing the user's hand. A Time-Of-Flight camera and an infrared camera system. This section discusses the advantages and disadvantages of the two different camera devices.

Infrared Camera

Inspired by the gesture pendant [7], we priorly started our investigations by the use of an infrared camera capturing an infrared illuminated gesturing hand. We found this solution to be limited due to the following reasons:

- The hand containing a relatively high albedo and being the closest object to the camera is expected to be the brightest object within the captured image. The hand's mask is isolated from the background by thresholding. The resulting mask contains only binary information and doesn't offer any additional information about the hand' pose and the positions of individual fingers. Therefore gestures which involve overlapping joints of our hand and the presented thumb on hand interactions of our input technology, are tremendously difficult to detect.

- The hole formed by the index finger and thumb, as the user performs the pinching gesture, is only visible to the camera if the hand is held in a specific angle. The user has to be aware of the camera's view at all times. Hence this detection system demands an inappropriate amount of attention for successful interaction.

- Algorithms to detect objects by the use of infrared cameras suffer from its ability to work in direct sunlight. The Sun floods every observed scene in a tremendous amount. Hence captured images of infrared cameras in direct sunlight contain no significant information about the hand's mask.

Depth Camera

Depth cameras using the Time-Of-Flight technology occupy a relatively new research field. The Time-Of-Flight concept represents a group of different technologies to determine the distance to an object by measuring the time the light requires to travel to the desired object. This concept is closely related to LIDAR scanners. Time-Of-Flight used for depth camera's enhance this solution by measuring multiple pixels within an image array. Hence it is possible to capture and analyse depth information of an observed field of view, even in real time. We strongly motivate the use of Time-of-Flight Technology for the detection of gestural input in mobile scenarios, because of the following features:

- Depth cameras capture detailed depth information of the observed area. This enables the reconstruction of 3D models representing the user's hand. Depending on the depth resolution of the capturing device, different levels of substantiation are thinkable. The resulting model can be used in order to implement rich interactions on top of the detected features of the hand. Further a three dimensional model of the hand, enables the user to not being forced to hold his hand within a precise pre defined angle in respect to the sensor. This a crucial property of the presented system and its eyes free usage.

- Depth cameras allow to isolate the region of interest from needless background information very easily. The company PMD [23] equipped their latest sensor chips with the SBI (Suppression of Background Illumination) technology, pushing this advantage even further. This solution enables a significant reduction of noise, caused by natural light flooding the observed area. Therefore current sensors are appropriate for mobile scenarios, since they can be used in every lightning condition of the user's changing environment. (see Figure 75).

- State of the art cameras offer the possibility to capture depth and grayscale information of the observed area simultaneously. This property realizes the opportunity to detect even more

precise features of the user's gesturing hand. Skin folds and creases are detectable depending on the camera's resolution and indicate advanced features of the user's hand.

Figure 75: Reconstruction of Hand in direct sunlight: Depth cameras are able to return the data of interest even under worst-case situations. In this case a hand is captured directly into the sun. The corresponding debug output image displays the hand's contour.

We found the sum of the mentioned properties of Time-Of-Flight Cameras as appropriate for the presented system. Yet the recent development within the research field of TOF cameras offer even more astonishing trends. The Camera, used for our implementation is a PMD 3.0 Camcube [23]. This camera offer the ability to measure the depth within a resolution of 5 mm and represented the state of the art in August 2010. Yet this resolution occupy a marginal edge for our purpose to detect detailed information of the hand. Another drawback of this camera system is its size, formed to bulky and heavy to be attached to the user's chest. Only a year later, cameras with the following astonishing features are released.

- A 60 degree field-of-view enables the user's hand to interact within a bigger space in front of his body.
- A 1 mm depth resolution offers the possibility to detect individual postures and bones of the hand.

- USB powered cameras, being smaller then a smartphone indicate mobile cameras to be available in near future.

A camera device containing all of the groundbreaking mentioned features is the PMD CamBoard [23]. The ongoing trend within the relevant fields of research, make it reasonable to assume that limitations of today's devices will be overcome.

7.3 DETECTION OF HAND AND THUMB INTERACTION BY TIME-OF-FLIGHT

The following chapter discusses our evaluation of tactile single handed input and our systems capabilities to detect those hand gestures. Our investigations priorly started by the vision to build algorithms, which reconstruct the three-dimensional model of our hand based upon the data of our point-cloud. This model was supposed to be a dynamic skeleton, which adapts the users hand pose as illustrated in Figure 76.

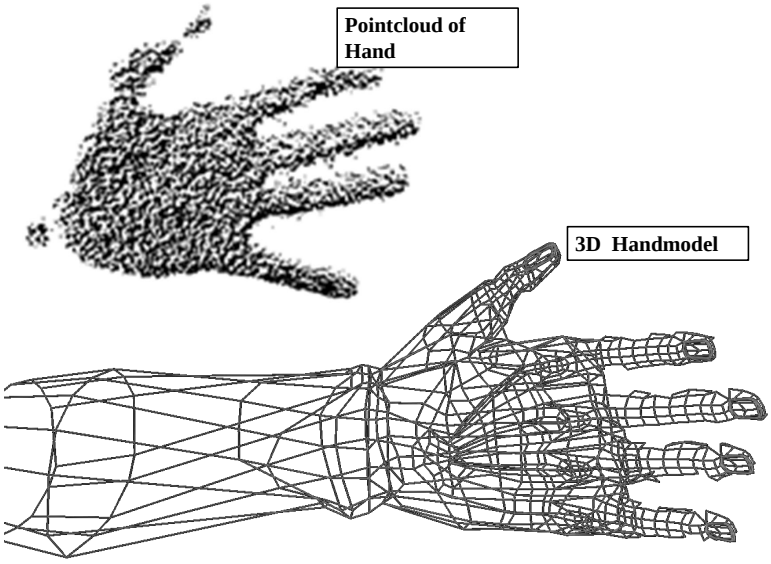

Figure 76: Model of hand skeleton reconstruction: Based upon the data of the captured point-cloud a three-dimensional model is reconstructed

.

Yet our investigations found this vision to be against impossible odds, due to the physical limitations of our hardware. The data captured by our chest mounted camera restricts the possibility of skeleton reconstruction, based upon the following properties:

- 2 1/2 Dimensions: The depth data of the captured environment, does not represent a full three-dimensional reconstruction of this area. Objects which are occluded by other objects in respect to the camera system, do not reflect infrared light back to the

camera lens. Hence there is no data indicating the position of those occluded objects. Additionally this constraint implicates that there is no information about an objects size in depth.

- Quality: The average depth noise of our camera system amounts only 0.5 cm. This amount of noise is outstanding nominal for contemporary devices, but still substantially strong for our purpose.

- Quantity: The captured image contains only forty-thousand pixel. A hand placed a half meter in distance to the depth camera occupy roughly about twenty thousand pixel within the image. The fingers and thumb, representing the actual region of interest, occupy only a fraction of those pixels. Additionally a tremendous amount of pixels are not correct or simply do not contain interesting information since hand segments are occluded by others. Hence the usage of statistical methods in order to improve the accuracy of the data is limited, by the lack of valid measurement samples.

Those limitations of our data merge together to a blockade. Therefore it is necessary to discuss the two major requirements of our system against each other.

- How much detail in detecting the interaction is the minimum for an efficient input concept.

- What is the maximum technically feasible of our system.

We propose an appropriate reduction of the desired features. Rather than detecting each finger separately, we can simply paraphrase all of our finger as a single input surface as illustrated in Figure 77 and detect the thumbs position in reference to the corresponding plane. This concept enables the possibility to realize a touchpad analogy. On that account the matching of each pixel to an individual finger becomes needless. Instead we use statistical methods on top of the sum of all pixels that belong to our finger, in order to approximate their arrangement in three dimensional space.

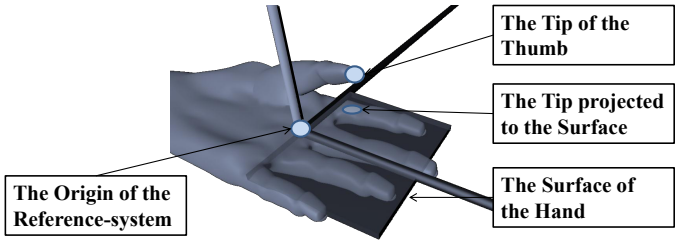

Figure 77: The hand as a touchpad: The thumb operates on top of an imaginary thumbpad fixed to the hand

In order to accomplish the desired task of feature detection, our depth camera system offered a marginal amount of depth noise. Hence the results of our solution revealed to much false detection to implement imaginary widgets. Since the investigation of widgets represent a major task, we decided to address this issue by in cooperating greyscale infrared measurements of the depth camera. This concept improved the stability of our system in a significant manner. We used the detection of shadows within the hand's mask, indicating invalid pixel's, in order to create more precise outlines and depth data of the hand. Yet this procedure is not advisable to be used in extreme conditions of natural light flooding the observed area. However the rapid development of today's depth cameras prove that this in cooperation of depth and infrared data, is no longer needed.

8

CONCLUSIONS

8.1 SUMMARY

Temporary mobile devices demand an inappropriate amount of the user's attention. We presented our concept of thumb on hand input for mobile devices in order to reduce this attention. Therefore we introduced a chest mounted camera system, which enabled the possibility to detect and analyse the user's gesturing hand by the use of computer vision. Exploring such a system our investigation found the following contributions as interesting:

- We found depth cameras using the Time-Of-Flight principle to be appropriate for detecting gesturing hand's in mobile scenarios.
- A robust preprocessor continuously analyses the observed stage in front of the user's body and successfully extracts the user's hand from unneeded background information.
- We described the detailed analyse of individual features of the user's hand, which expand common possibilities of detecting gestural input by optical solutions.
- We introduced imaginary widgets, representing rich interaction types by the use of thumb and hand interplay.
- We described sample applications, which are controlled by imaginary widgets and demonstrate the efficiency of our solution.

8.2 future work

There are many interesting prospects for the presented gestural input system. We found our system to be promising and encouraging. Yet it still suffers from drawbacks which should be investigated in future works:

- Push the level of detail: We are confident about the efficiency of our widgets. Still we believe that the presented system offer even more possibilities to interact in mobile scenarios. Hence advanced algorithms detecting even more precise features of the hand could enable more detailed types of interactions like twirling the thumb around individual fingers and assigning items to different fragments within each finger.
- Investigation of a complete prototype, demonstrating the interaction with the most essential functions of a smart phone represent a very interesting task.
- User studies about the humans ability to intuitively interact with the proposed technology. In particular it would be interesting to investigate the abilities to memorize type and position of individual widgets. Additionally we found circular motions by the tip of the thumb to be way harder performed then linear motions like sliders. Corresponding studies in order to evaluate and compare the efficiency of different imaginary widgets are appropriate.
- Feedback is a crucial task for imaginary widgets. Research about appropriate feedback systems are interesting and much needed in order to proceed.

Our implementation consists mainly of statistical methods, approximating three dimensional orientation of the hand, combined with conditional expressions checking for the existence of pre defined assumptions. Those expressions need to be replaced by complex machine learning constructs using a great number of training data. The resulting algorithms could possibly lead to the desired full skeleton

model of the hand, given that the three dimensional data captured by depth cameras increase their level of detail in near future. This solution is much needed and could set further foundations to investigate the efficiency and usability of the presented system.

BIBLIOGRAPHY

[1] Thomas Baudel and Michel Beaudouin-Lafon. Charade: remote control of objects using free-hand gestures. *Commun. ACM*, 36:28–35, July 1993.

[2] Doug A. Bowman, Chadwick A. Wingrave, Joshua M. Campbell, and Vinh Q. Ly. Using pinch gloves for both natural and abstract interaction techniques in virtual environments. In *Human-Computer Interaction*.

[3] Lars Bretzner, Bj"orn Eiderb"ack, and S"oren Lenman. Computer vision based recognition of hand gestures for human-computer interaction. Technical report, 2002.

[4] Wiimote controller and sensor. Nintendo. Technical report, http://www.nintendo.com/wii, 2005.

[5] Yanqing Cui, Jan Chipchase, and Fumiko Ichikawa. A cross culture study on phone carrying and physical personalization. In *Human-Computer Interaction*, pages 483–492, 2007.

[6] J. eric townsend. Powerglove. Technical report, http://mellottsvrpage.com/glove.htm, 1989.

[7] Maribeth Gandy, Thad Starner, Jake Auxier, and Daniel Ashbrook. The gesture pendant: A self-illuminating, wearable, infrared computer vision system for home automation control and medical monitoring. In *Proceedings of the 4th IEEE International Symposium on Wearable Computers*, ISWC '00, pages 87–, Washington, DC, USA, 2000. IEEE Computer Society.

[8] P5 Glove. Essential reality. Technical report, http://www.vrealities.com/P5.html, 2002.

[9] Sean Gustafson, Daniel Bierwirth, and Patrick Baudisch. Imaginary interfaces: spatial interaction with empty hands and without visual feedback. In *User Interface Software and Technology*, pages 3–12, 2009.

[10] Chris Harrison, Desney Tan, and Dan Morris. Skinput: appropriating the body as an input surface. In *Proceedings of the 28th international conference on Human factors in computing systems*, CHI '10, pages 453–462, New York, NY, USA, 2010. ACM.

[11] Sanna Kallio, Juha Kela, Jani Mäntyjärvi, and Johan Plomp. Visualization of hand gestures for pervasive computing environments. In *Proceedings of the working conference on Advanced visual interfaces*, AVI '06, pages 480–483, New York, NY, USA, 2006. ACM.

[12] Juha Kela, Panu Korpipää, Jani Mäntyjärvi, Sanna Kallio, Giuseppe Savino, Luca Jozzo, and Sergio Di Marca. Accelerometer-based gesture control for a design environment. *Personal and Ubiquitous Computing*, 10(5):285–299, 2006.

[13] Luv Kohli and Mary Whitton. The haptic hand: providing user interface feedback with the non-dominant hand in virtual environments. In *Proceedings of Graphics Interface 2005*, GI '05, pages 1–8, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2005. Canadian Human-Computer Communications Society.

[14] A. Kolb, E. Barth, R. Koch, and R. Larsen. Time-of-flight cameras in computer graphics. In *Computer Graphics Forum*, pages 141–159, 2010.

[15] Steinar Kristoffersen and Fredrik Ljungberg. Making place to make it work: empirical explorations of hci for mobile cscw. In *Proceedings of the international ACM SIGGROUP conference on Supporting group work*, GROUP '99, pages 276–285, New York, NY, USA, 1999. ACM.

[16] Playstation Move. Sony playstation. Technical report, http://us.playstation.com/ps3/playstation-move, 2009.

[17] Tao Ni and Patrick Baudisch. Disappearing mobile devices. In *User Interface Software and Technology*, pages 101–110, 2009.

[18] Tao Ni, R. P. McMahan, and D. A. Bowman. Tech-note: rapmenu: Remote menu selection using freehand gestural input. In *Proceedings of the 2008 IEEE Symposium on 3D User Interfaces*, 3DUI '08, pages 55–58, Washington, DC, USA, 2008. IEEE Computer Society.

[19] Shwetak N. Patel, Julie A. Kientz, Gillian R. Hayes, Sooraj Bhat, and Gregory D. Abowd. Farther than you may think: An empirical investigation of the proximity of users to their mobile phones. In *Ubiquitous Computing/Handheld and Ubiquitous Computing*, pages 123–140.

[20] J. Sklansky. Measuring concavity on a rectangular mosaic. *IEEE Transactions on Computers*, 21:1355–1364, 1972.

[21] Thad Starner, Alex Pentland, and Joshua Weaver. Real-time american sign language recognition using desk and wearable computer based video. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20:1371–1375, December 1998.

[22] Thad E. Starner, Cornelis M. Snoeck, Benjamin A. Wong, and R. Martin Mcguire. Use of mobile appointment scheduling devices. In *Computer Human Interaction*, pages 1501–1504, 2004.

[23] PMD Technology. Pmd vision cameras. Technical report, http://www.pmdtec.com/, 2011.

[24] Andrew D. Wilson. Robust computer vision-based detection of pinching for one and two-handed gesture input. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*, UIST '06, pages 255–258, New York, NY, USA, 2006. ACM.

[25] Catherine G. Wolf. Can people use gesture commands? *SIGCHI Bull.*, 18:73–74, October 1986.

[26] Thomas G. Zimmerman, Jaron Lanier, Chuck Blanchard, Steve Bryson, and Young Harvill. A hand gesture interface device. *SIGCHI Bull.*, 17:189–192, May 1986.

LIST OF FIGURES