A CAMERA-BASED HAND-GESTURE DETECTION SYSTEM FOR INTERACTING WITH EYEWEAR COMPUTERS

Master's Thesis
IT University of Copenhagen
October, 2014

Author:
Daniel Lujan Villarreal
dluj@itu.dk

Supervisor:
Thomas Pederson
tped@itu.dk

Abstract

In the recent years, wearable computing has gained attention with the new eyewear computing devices like Google Glass. Voice commands, one-handed keyboards, and smartphones have been used to provide input to the wearable devices, but these have their own limitations, and interaction with eyewear computers is still a challenge. We argue that hand gestures can provide a natural and intuitive way to provide input to eyewear computers.

We develop a prototype that supports hand-based gestures as input modality for eyewear computers. We define a set of gestures and develop a hand-based gesture detection method optimized for our set of gestures that relies on the characteristics of the hand.

Our results show that our gestures are intuitive, most of them ergonomic, easy to remember and lack ambiguity. As a complementary measurement, the results show a precision of 96.91% and a recall of 89.52% of the gesture detection method.

We also discuss possible improvements for our prototype. Moreover, as a result of our observations through the design process of the prototype and the result of the evaluations, we present a set of design guidelines useful for future systems supporting hand-based gestures as input modality.

Contents

1 Introduction

By emerging new eyewear computing devices such as Google Glass and Vuzix Smart Glass, wearable computing has gained widespread attention, and we expect that in the close future people use new eyewear computing devices in everyday tasks.

The conventional methods of interaction, like keyboard and mouse, assume a fixed physical relationship between user and device (Siewiorek, 2002). This methods do not apply anymore since they would considerably reduce the efficiency of the wearable system. New methods of interaction with wearables are provided by voice commands, hand-held controllers, and even smartphones. However, these interaction techniques still present challenges and limitations.

Humans naturally use gestures and point with their hands while communicating with others (Hinckley, 2007). Gesture based interaction has gained attention in consumer electronics with game consoles like Nintendo Wii[1] and Microsoft Kinect[2]. Hand gestures as input, or touch less interaction, has gained attention in the research community over the years because it removes the physical barrier with an interactive system. Another advantage of touch less interaction is maintaining a sterile environment because direct contact with objects is not required (Barré et al., 2009), a feature highly important for Hospitals.

We believe that hand gestures can provide a suitable, intuitive, and natural interaction technique for eyewear computers. The main idea in this thesis is to study hand-based gestures as input modality.

1.1 Problem Statement

Historically, voice commands, body gesture commands, one-handed keyboards (such as Twiddler [3]), hand-held controllers, and even smartphones have been used to provide input to the wearable computers. But these conventional modalities have their own limitations, and there is a growing need for new intuitive interaction techniques for wearable devices. Hand-held controllers and smartphones are external to the eyewear computers and often inconvenient and undesirable in free-form interaction scenarios and mobile settings. The accuracy of voice recognition varies form user to user and is susceptible to ambient noise (Colaço et al., 2013). Wearables should offer seamless integration of information processing tools with the environment. In order to accomplish this, they must offer functionality in a natural, intuitive, and unobtrusive manner (Siewiorek, 2002). A typical eyewear-computing device includes a head-mounted display, a small processing unit, and a HD camera. This setting enables the wearable device to support hand gestures as input modality.

1.1.1 Goal

The goal of this master thesis is to study hand gestures as input modality for eyewear computers. We will develop a hand based gesture recognition system for an eyewear computer using its camera. In order to implement the system we will define a set of gestures that are intuitive to the user and implement a gesture detection method optimized to detect this set of gestures. An user evaluation will be held where the user follows a scenario using the prototype. The intuitiveness of the gestures and the user experience will be evaluated with a questionnaire. We do not seek a novel gesture detection method, thus the precision and recall of the gesture recognition will be evaluated as a complementary measurement in ideal light conditions and uncluttered background.

[1] http://www.nintendo.com/wiiu

[2] http://www.microsoft.com/en-us/kinectforwindows/

[3] http://twiddler.tekgear.com/

1.1.2 Method

In this project we follow an iterative design process. We start with a brainstorming session that will lead to the definition of the set of gestures. Based on the definition of the gestures we design and implement the gesture detection method. In the next step, a series of iterations follow with formative evaluations and new versions of the prototype. Finally, we design and conduct a user evaluation where we evaluate the intuitiveness of the gestures, the user experience, and the precision and recall of the recognition method.

1.2 Use scenario

The motivation for our scenario comes from the application area our supervisor and his PhD student are focusing on at the time of writing, e.g. Hospitals. One of the main characteristics of hospitals is mobility. They are busy places with constant movement of patient beds, records, and x-rays (Bardram and Bossen, 2005). Also, sterile environments are common in hospitals. In the operating room, a special nurse is available only to operate the computer if the surgeon wants to take a closer look at an x-ray. The surgeon cannot touch the computer for fear of contamination (Bardram et al., 2006). A mobile and wearable solution can help to decrease the mobility of the staff and the physical interaction with computers.

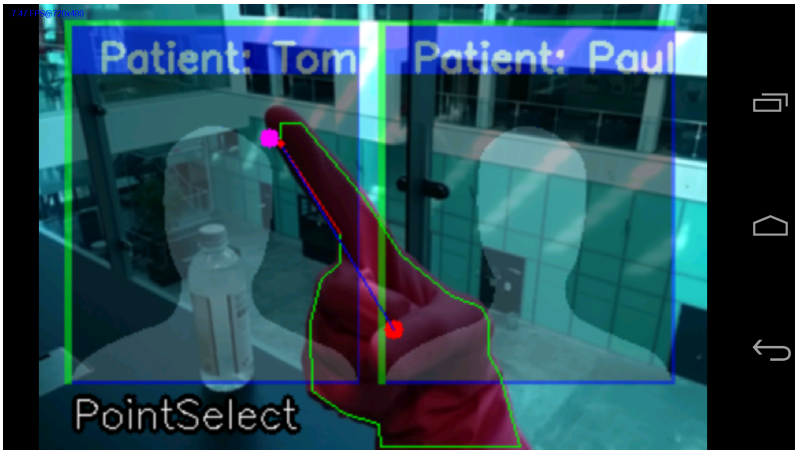
For the definition of the scenario we referred to Bardram and Bossen (2005), Carlsson et al. (2006), and a yet unpublished ethnographical study by Shahram Jalaliniya in a Danish Hospital. From the previous studies we abstract the following activities of a surgeon during ward rounds. Firstly, the surgeon reviews the patient's record. Secondly, the surgeon interacts with the patient, e.g. for further examinations or simply to talk. Finally, decisions about the further treatment are done. We define our scenario based on the surgeon's first activity.

The patient's record contains important information, e.g. current status, x-rays pictures, and laboratory results. If the record is printed beforehand the surgeon can go directly to the patient's bed. On the other hand, if the record is not available in print, or not up to date, the surgeon has to go to a special room and look for the information in a computer. In both cases the surgeon loses time either in reading the record by the patient's bed or in going to the computer room. In our scenario, the surgeon reviews the patient's record with our system, here described as assistant glasses, while visiting the patient.

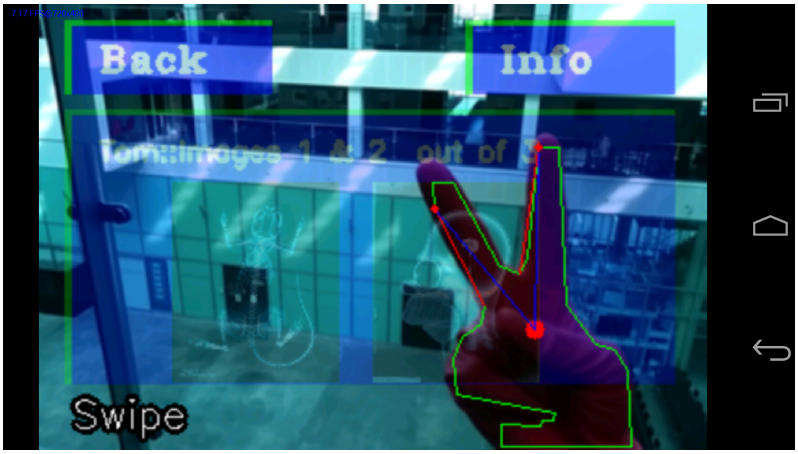
> *Robert, a surgeon, is doing a ward round to check on several patients. When he arrives to his next patient, Tom, the assistant glasses are waiting for Robert to select the patient's profile being visited (figure 1a).*
> *Robert selects Tom as the patient being visited and the assistant glasses display Tom's record on screen. Robert selects the Images button (1) to review the available x-rays (figure 1b) and performs the Swipe gesture (2) to browse through the x-rays and finally selects x-ray number three (1). After the selection, the assistant glasses display the x-ray in a bigger size (figure 1c). He wants to focus on a smaller area of the x-ray so he zooms in the picture (3). After reviewing this x-ray Robert focus his attention on the patient and stops interacting with the assistant glasses. After several seconds the assistant glasses assume the interaction session is over.*
> *Robert decides that he wants to review x-ray number two so he starts re-starts the interaction with the assistant glasses (4) and clicks the back button (1) to go back to the available x-rays and selects number two (1). He then rotates the x-ray (5) and zooms in (3) to get a better perspective of the picture. While zoomed-in,*

Robert decides to focus on a different area of the x-ray, he selects (1) the new point to be in the center of the displayed image. In the middle of selecting a new focus area Robert decides to leave the displayed image as it is and stops the current interaction (6).
After reviewing the record Robert congratulates Tom for his recovery and resumes to other activities.

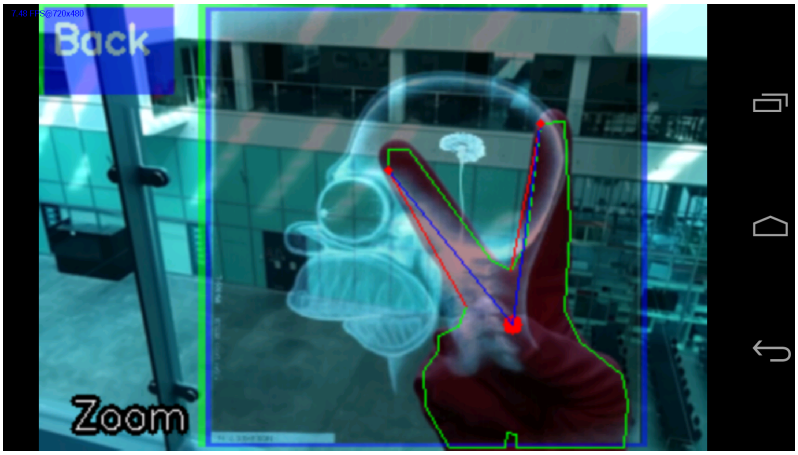
The numbers (1-6) regard the gestures supported by the system and their scope. The previous scenario describes how the system can be useful in the work place. The surgeon would not need to go to a special room or have physical contact with any computers in order to review the patient's record. The latter also enables the system to also be suitable for use in an operating room in the future.



(a) PatientSelection



(b) RecordViewing



(c) ImageInteraction

Figure 1: Early stages of the prototype

2 Related work

The goal of this thesis is the study of hand gestures as input modality for eyewear computers. The outcome of this study falls in *Human-Computer Interaction* (HCI), defining our first research area. In order to accomplish the study we need to develop a prototype that supports this interaction technique. Although it is not within our scope to contribute outside HCI, the challenges we face in the implementation of the prototype cover the areas of *Gesture Detection* and *Wearable Computing*, and we define these two as our second and third research areas. We position ourselves in the overlap of these research areas (figure 2). With our research area defined we start by looking at similar systems, how we relate to them, and how we differ from them.

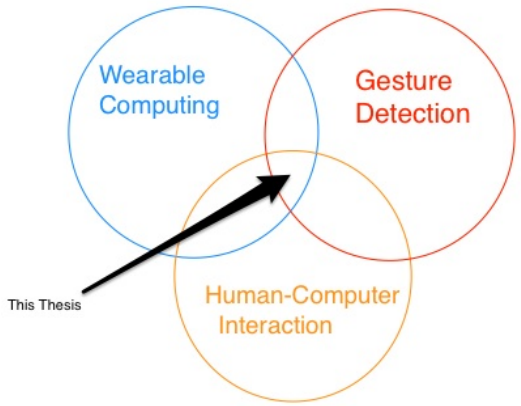


Figure 2: Overlapping research areas of this thesis.

The systems of our interest have the characteristics of 1) making use of hand or arm gestures as input modality, and 2) are developed for wearable devices. Our research gave us many results, so we break them into two groups. The first group consist of systems aided by wearable devices. The second group consist of systems that only rely on a camera for the detection of gestures. Within these two groups there are several popular techniques that are proven to be effective in gestures recognition. The rest of the chapter is as follows. Firstly, we describe two techniques that have been successful in gesture recognition. Secondly, we address systems that recognize hand or arm gestures aided by wearable devices, e.g. sensors in smart watches or worn IR cameras. Thirdly, we describe systems that only rely on a 2D camera for the detection of hand gestures, e.g. a 2D camera in an eyewear computer. Finally, we summarize and discuss these systems in relation to our work.

2.1 Techniques for gesture recognition

Mitra and Acharya (2007) presents a survey of the tools for gesture recognition that have been proven successful. In this section, we address two popular techniques used for gesture recognition.

2.1.1 Hidden Markov models

Hidden Markov Models, from now on HMM, are characterized by their learning ability. They are presented with time-sequential data and automatically optimize a model with the data. HMMs consist of states with a probability of transition from one state to another, and these states depend only on the state at the preceding time. Transitions between states occur stochastically. HMM states

are not directly observable, and can be observed only through a sequence of observed symbols. One HMM is created for each gesture.

In order to recognize gestures with HMMs, each frame in the time sequential images is transformed into feature vectors. The feature vectors are then clustered by a pattern classification technique. The resulting clusters are transformed into a sequence of symbols, which are used for the recognition and training phase (Yamato et al., 1992).

The global structure of HMMs has parallel connections of each HMM, thus adding or removing new gestures can be easily done (Mitra and Acharya, 2007).

2.1.2 Finite state machines

In the Finite state machine approach, FSM from now on, a gesture can be modeled as an ordered sequence of states in a spatial temporal space. The number of states in the FSM varies between applications depending on the implementation and number of gestures supported. A very nice feature of FSMs is that their state-based representation can be extended if more gestures are to be supported.

The gestures are usually represented by set of points, e.g. positing of the head or hand, in a 2D space. These sets of points constitute the gestures models. Sensor data, e.g. image sequence from the camera, is acquired to be compare to the models and recognize the gestures.

The training of the models is usually done off-line with several examples of the gesture, and the recognition of the gestures is performed online using the trained FSM. The gesture recognizer decides to either stay at the current state of the FSM or jump to the next step based on the given input data. When the FSM reaches its final state a gesture has been recognized. In the event that more than one FSM recognizes a gesture, there is usually a winning criteria to determine the gesture that will be recognized (Mitra and Acharya, 2007).

2.2 Gesture recognition based on sensors and wearable devices

In this subsection we describe systems that support gesture recognition based on sensors and wearable devices. With these systems we can get an overview of the different input devices, feedback methods, design factors, and information needed in order to support gesture detection as interaction technique. We do not restrict this part to only be hand-based gestures since we can take advantage of the different methods and algorithms used for gesture detection based on other parts of the body or movement.

GestureWrist and GesturePad: Unobtrusive Wearable Interaction Devices (Rekimoto, 2001). The GestureWrist and GesturePad are input devices for wearable computers that allow interaction between users and wearable computers using gesture-based commands. The GestureWrist is a device in the form of a wristband that can detect hand gestures and forearm movements by measuring changes in the wrist shape with sensors embedded in the wrist band. The GesturePad is a sensing module that can be attached to the user's clothes. The interaction with the GesturePad can be done from the outside of the clothes even if the module is placed on the inside. The GesturePad can detect when fingers are close to it and calculate their position.

The hand gestures are recognized with capacitive sensing by measuring the changes of the arm shape on the inside of the wristband (see figure 3). The gestures are defined as the combination of a hand shape and arm positions. The hand shapes can be making a fist or pointing. The different arm positions are palm up, palm right, palm left, palm down, forearm up, and forearm down. The hand shapes separate gesture commands into segments, and two arm positions make up one input command (e.g. palm left and palm down).

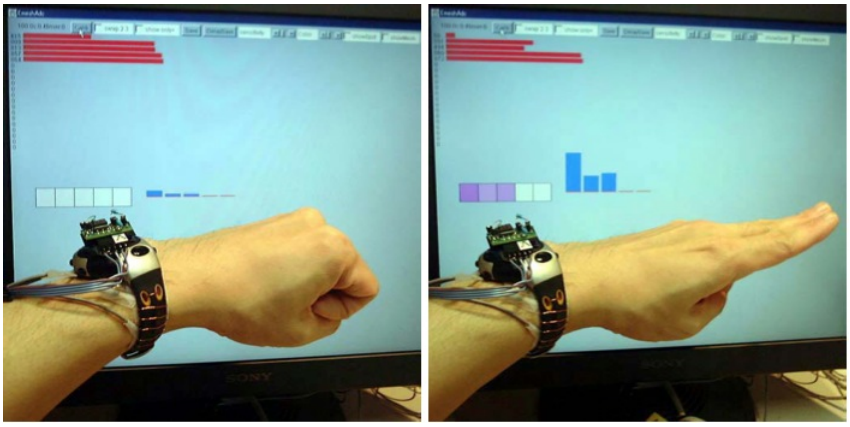


Figure 3: GestureWrist and GesturePad: Unobtrusive Wearable Interaction Devices

In this work they focus on supporting two features that would make wearable computers to be used in everyday life situations. The first one is the support of hands-free operations and allowing the user to quickly switch between normal and operation modes. The second is social acceptance. They mention that input devices should be as natural as possible for them to be used in different social situations. They believe ”unobtrusiveness” is essential for a wearable device to be widely accepted by the public.

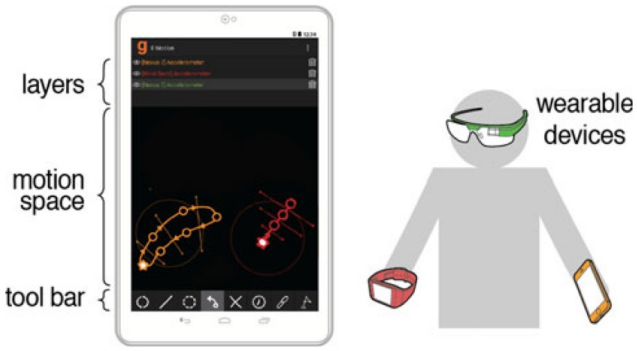
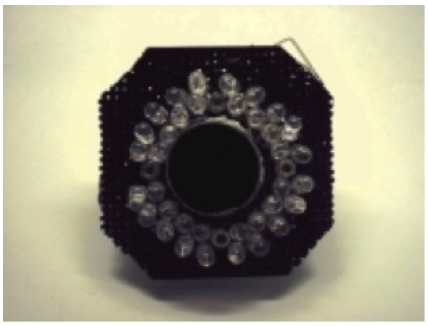


Figure 4: Interface of CompositeGesture and an example set of devices.

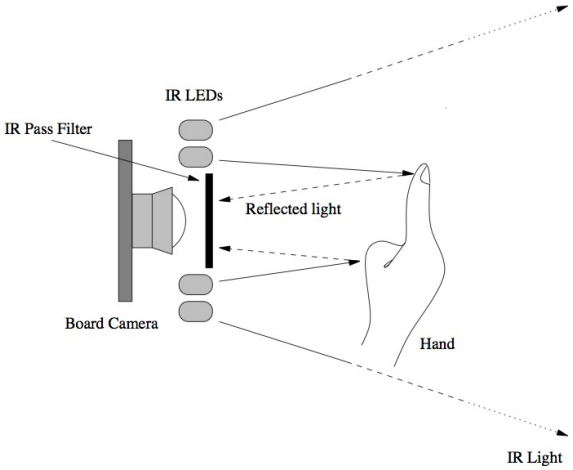
CompositeGesture: Creating Custom Gesture Interfaces with Multiple Mobile or Wearable Devices (Kim et al., 2014). Composite Gesture is a mobile application that allows a user to customize gestures using motion sensors form multiple mobile or wearable devices. In other words, a user can define a gesture by shaking his smart watch horizontally. A very interesting feedback method is also presented. They define a motion space fed by the accelerometers of the device. With a glass-box approach, the user can define a gesture and see what they are creating in an interface showing this motion space (figure 4). Also, with the ability of the user to define gestures, they present a very good approach to the recognition of natural and intuitive interaction.

They evaluate their application with four participants. The sessions lasted 30 minutes and the participants were given a watch-type device and a mobile phone. An interview followed the session

where usability problems and potential applications were discussed. Their results are that the application was successfully used and also uncover some usability aspects. One of the usability aspects they discuss are that even though the participants were motivated and willing to spend their time customizing gestures, they preferred a predefined gesture set for things they were not engaged in. For example, turning on a television set.



(a) The pendant.



(b) Sideview of pendant with IR reflecting off the user?s hand.

Figure 5: The Gesture Pendant: A Self-illuminating, Wearable, Infrared Computer Vision System for Home Automation Control and Medical Monitoring.

The Gesture Pendant: A Self-illuminating, Wearable, Infrared Computer Vision System for Home Automation Control and Medical Monitoring (Gandy et al., 2000). The gesture pendant is a wearable device that allows the user to control the home automation system Aware Home(Kidd et al., 1999) with hand gestures. The gesture pendant is worn as a part of a necklace or a pin. It has cameras and infrared LEDs with a pass filter over the lens. This setup makes it capable of tracking the hand in the dark as well. The start of the interaction with the pendant is controlled by a button.

The recognition of the gestures is done with image processing in a desktop computer. Two types of gestures can be recognized. The first are control gestures which provide continuous output while being detected. These are detected with pattern recognition using the nearest neighbor algorithm. The second are gestures defined by the user which provide discrete output when detected. In other words, a control gesture can be to turn down the volume of the T.V. and a user defined gesture can be to turn on or off the T.V. The user defined gestures are recognized with HMMs.

The set of control gestures defined are 1) vertical pointed finger (vf) 2) horizontal pointed finger (hf) 3) horizontal flat hand (hfh) and 4) open palm (op). Six gestures were trained and testes as the user defined gestures. These included 1) fire on 2) fire off 3) door open 4) door close 5) window up and 6) window down. Their results show an accuracy of 96.67% for the HMMs and a 95% correct classification of the gestures with the One-Nearest Neighbor algorithm.

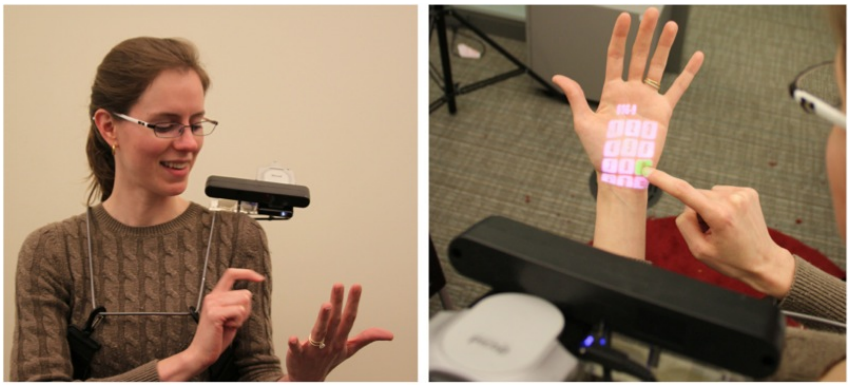


Figure 6: Projection of OmniTouch on the palm of the hand. To the right the click action is shown.

OmniTouch (Harrison et al., 2011). This is a wearable depth-sensing and projection system that enables interactive multitouch applications on any surface. That means that any wall, table, hand, chair, board, and object in general can be used as a surface to interact with the system. It provides capabilities similar to a mouse or touchscreen, like X and Y coordinates, and can differentiate between hovering an object and clicking it. The projection of OmniTouch is always hand-sized as seen of figure 6, even when projecting in big surfaces. This is based on the idea that the projection can only be as big as the smallest conceivable surface, e.g. the hand. It uses an interesting algorithm to detect click or hover actions using flood fill towards the fingertip on the depth map. Possible applications for OmniTouch vary widely because of its capabilities. With the option of click and drag, buttoned and scrollable interfaces are an immediate possibility. One possible application of the system described in this work is a phone keypad with a ”slide to unlock” feature. They performed a user study to evaluate and demonstrate their approach. Their results include a 96.5% performance of the finger click detection.

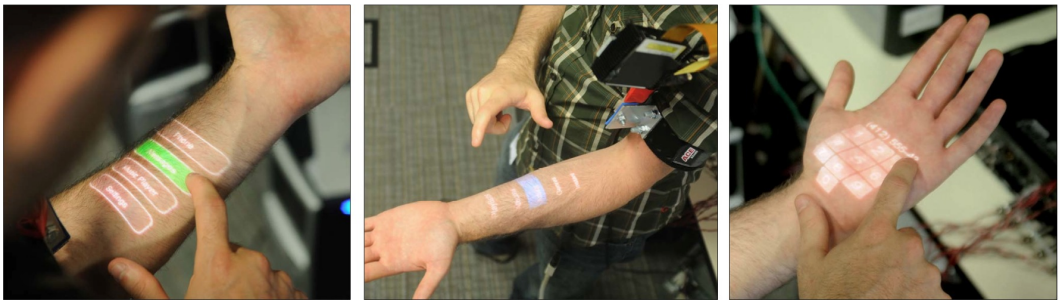


Figure 7: Skinput sensing armband augmented with a pico projector. Here we can see some possibilities for the system.

Skinput: Appropriating the Body as an Input Surface (Harrison et al., 2010). Skinput provides a very interesting approach for the detection of gestures using the body. The main idea behind this approach is that when a user taps its body a mechanical vibration is propagated and can be detected using an array of sensors placed on an armband. The motivation for this technique is that devices with significant computational power can now be easily carried on our bodies. However, they argue that because of their size the interaction space tends to be limited affecting the usability

and functionality of the devices. Thus, appropriating the body as an input surface gives the user extra space for this interaction. Also, most of the body can be reached by the hands making it a suitable input surface with many possibilities. Although the whole body can be used as the surface, in this work they focus on the arm, as seen on figure 7, giving an continuous flat projective surface when combining the armband with a pico projector.

They use a brute force machine learning approach combining 186 features. These features are passed to a Support Vector Machine, or SVM. In order to train the SVM, a collection of several examples of each input location of interest is acquired.

The data for the recognition is processed in an external PC. Here, the received audio stream is segmented into individual taps with the same features of the SVM. After the segmentation, the waveforms are sent to the SVM and analyzed for classification. After the input is classified, an event associated with the location is instantiated.

The performance of the system is evaluated in an experiment. The results of this experiment show that the accuracy of the classification is 87.6%.

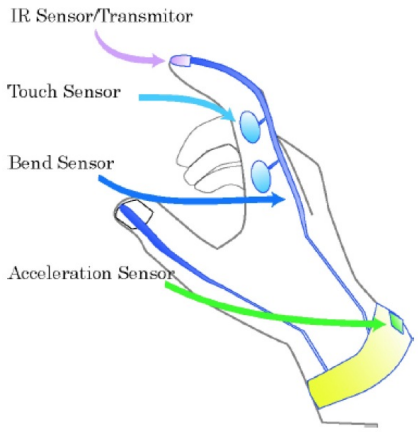


Figure 8: Ubi-Finger visual representation. Here we can see the arrangement of sensors and transmitter used by the wearable device.

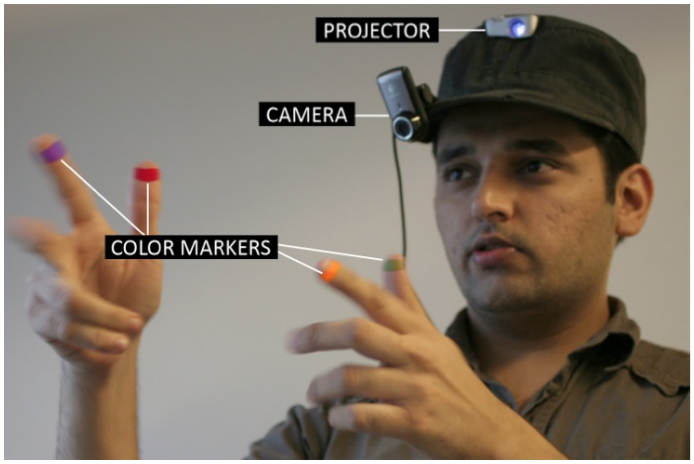
Ubi-Finger: Gesture Input Device for Mobile Use (Tsukada and Yasumura, 2002). Ubi-Finger is a wearable device that acts as an interface for appliances surrounding the user. It has three main concepts 1) friendly operations with natural gestures, 2) optimized for mobile use, and 3) multiple uses with a common interface. As seen on figure 8, the device places sensors on the hand to recognize finger gestures. The system proposes a very interesting way of interaction with appliances around the user. Ubi-Finger has the ability to control appliances by pointing at them. This is made possible by the complete arrangement of the system which includes the 1) Ubi-Finger, 2) appliances connected to the network with a unique ID, 3) a host computer to recognize the gestures, and 4) a server that maps the IDs of the devices with the ID of the user or operating commands.

The overall idea of the interaction with appliances is the following. When the user points Ubi-Finger at an appliance its ID is associated in the server. When a gesture is detected by the host computer the server is notified and proceeds to send a command to the selected appliance. In other words, the user can point to the light controlling appliance (assuming it is digital and connected to the network) and turn it off by lowering the hand.

The gestures are detected with the bending degree of the index finger and tilt angles of the wrist. The gesture recognition can be started or stopped with a trigger mechanism, which is touching the

touch sensor with the thumb.

They evaluate the system in an experiment with 10 participants. The experiment consisted of controlling multiple appliances with the prototype of Ubi-Finger. After each session, the participants answered a questionnaire to get their subjective feedbacks.



(a) Here, we can see the wearable camera, projector, and the finger's color markers .

(b) Here, we can see an example of the Map application using the wall as the display surface.

Figure 9: WUW.

WUW - Wear Ur World - A Wearable Gestural Interface (Mistry et al., 2009). This paper presents WUW, a gestural information interface. WUW consists of a camera and a small projector. As we can see in figure 9a both devices can be attached to everyday clothing. The projector visually augments the surfaces of projection, while the camera is used for the detection of the gestures. This setup enables the surfaces around the user to be used for the projection. The gesture detection is aided by colored marks on the fingers. Several proof-of-concept applications are developed to

demonstrate the use cases for this interface. In figure 9b we can see the Map application, which allows the user to navigate through a map displayed on the wall.

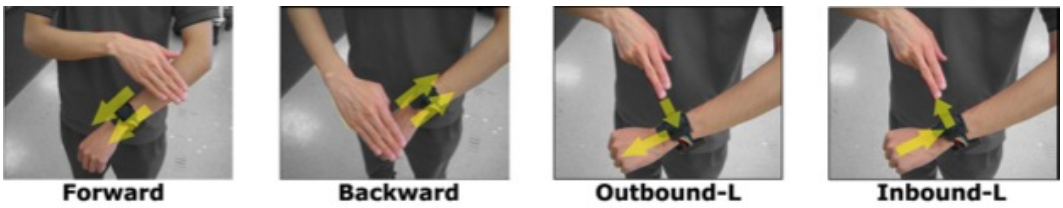


Figure 10: AirTouch: Synchronizing in-air hand gesture and on-body tactile feedback to augment mobile gesture interaction.

AirTouch: Synchronizing in-air hand gesture and on-body tactile feedback to augment mobile gesture interaction (Lee et al., 2011). This work presents an improvement of a previous project implemented by the same team, GestureWatch (Kim et al., 2007). AirTouch is a wristwatch interface that supports gesture interaction through tactile feedback and a PTG (push-to-gesture) mechanism which allows the user to perform a gesture and then confirm it with a trigger gesture. The GestureWatch has proximity sensors with vibration motors allowing the tactile feedback. When the hand of the user comes near the watch the corresponding motor buzzes as a feedback method for the user to know that the hand was acknowledged. Once the user is certain that the correct gesture was detected the confirmation has to be done. The trigger gesture is defined by the wrist being lifted towards the sky and back to its natural position. If this trigger gesture is not performed, the previously detected gesture is canceled. The gestures supported by the system are hand movements above the wristwatch. For example, moving the hand above the wristwatch from the elbow towards the tip of the fingers would detect the forward gesture. The detection of the gestures is done with GART toolkit (Lyons et al., 2007), which uses HMM.

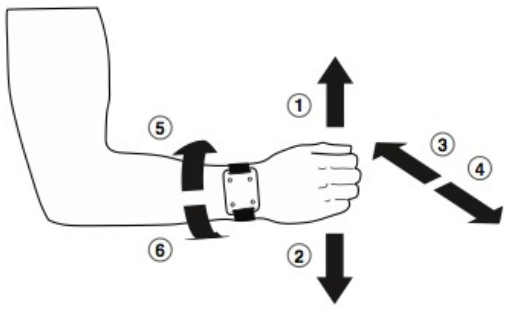


Figure 11: Gestures supported in Touch-less Interaction with Medical Images Using Hand & Foot Gestures.

Touch-less Interaction with Medical Images Using Hand & Foot Gestures(Jalaliniya et al., 2013). This work presents a combination of hand and foot gestures to interact with images. It is based on a wristband with sensors for the hand gestures and capacitive floor sensors for the foot gestures. The wristband contains different sensors, like accelerometer and gyroscope, that allow for the classification of the gestures. The gestures are defined by the movement of the hand and arm.

The classification of gestures is done with a two phases approach. The first phase uses a single neural network to differentiate between 1) the hand being idle and 2) the rest of the gestures. The second phase classifies the gesture according to predefined models using six neural networks. A prototype was implemented to demonstrate the method. In this prototype a surgeon interacts with medical images in the operating room. Six gestures are supported. The first four are a set of regular arm movements (up, down, right, and left) with the purpose of menu navigation. The remainder are a set of 2 gestures for zooming in and out the pictures. The foot gestures have two purposes. The first is to enable or disable the interaction with the displays and the second is to switch between screens.

2.3 Gesture recognition based on camera

In this section we present related systems that support gesture interaction based on camera. Here, we pay closer attention to hand-based gestures. These systems provide a closer overview of the different algorithms and techniques for hand gesture detection. We take a closer look to challenges like detection of dynamic gestures, light conditions, and intuitiveness in the gesture vocabulary.

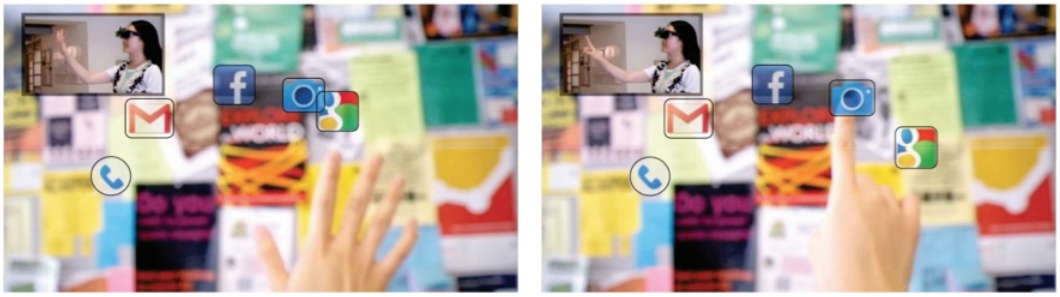


Figure 12: Example of application for MIME. An augmented menu for navigation.

Mime: Compact, Low-Power 3D Gesture Sensing for Interaction with Head-Mounted Displays (Colaço et al., 2013). MIME is a 3D sensor that supports interaction with head-mounted displays. The interaction is designed to be short ranged and single handed. The sensor contains an array of three photodiodes that triangulate the time of flight of a pulse LED. With this triangulation, it localizes the hand coordinates to define a region of interest, or ROI, on the RGB image. The ROI is further processed using skin color segmentation, shape recognition, feature extraction, and finally the gestures are recognized using a trained SVM classifier. This combination of time of flight 3D sensing and RGB image gives a robust detection of gestures even in difficult light conditions. The use of 3D sensing gives MIME the possibility to define different types of gestures. For example, they define the zoom gesture as moving the hand towards and away from the screen. Also, the clicking action is made more natural as a ”pushing” gesture. Some prototypes are presented to demonstrate the concept. For example, an augmented menu for navigation in an augmented interface (figure 12), or an application for in-air drawing.

Vision Based Interpretation Of Hand Gestures (Argyros and Lourakis, 2006). In this paper a Vision-based interface is presented for the controlling of a computer mouse via hand gestures. The gesture detection is a previous work by the same team that permits the detection and tracking of multiple hands moving freely in the field of view of a camera. The system has the ability to

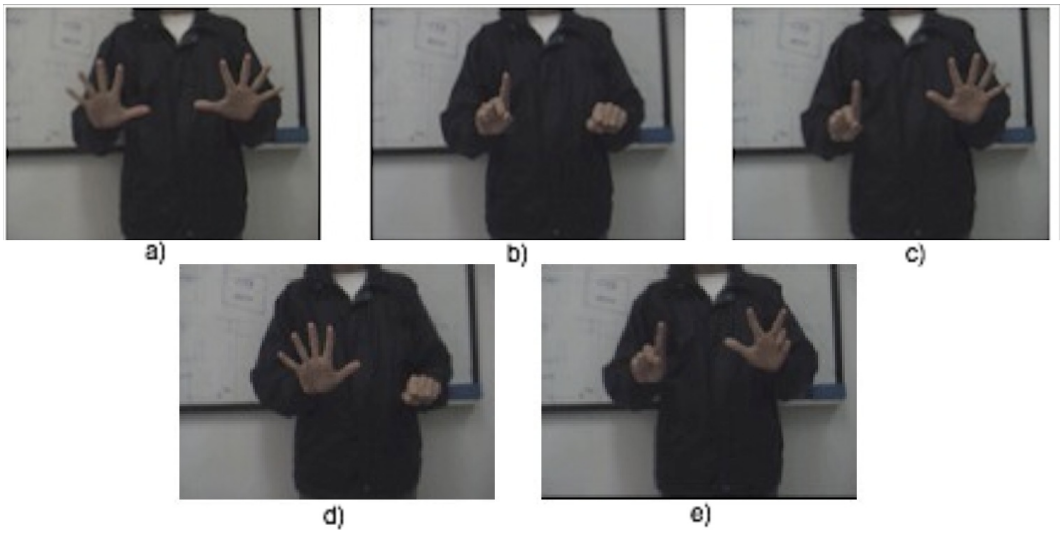


Figure 13: Vision Based Interpretation Of Hand Gestures. Here we can see representative instances of the gestures a) activate and deactivate gesture-based mouse control, b) move mouse, c) left button click, d) right button click, and e) left button double click.

dynamically adapt to illumination changes. Two types of hand gestures are studied. The first type is based on a camera using 2D information. The second type relies on 3D information and requires stereoscopic input. The gestures detected by the 2D approach are static and require the two hands. The gesture vocabulary consist in part of mouse control activation and deactivation, and several operations with the mouse like left click or left button pressed. The second approach consists of the same gestures as the first approach adding depth information and two dynamic gestures. The depth information allows the gestures to be performed only with one hand. The first dynamic gesture is the left mouse button where the closed hand is extended towards the camera. The second dynamic gesture is the opposite as the first one where the closed hand retracts from the camera, meaning release left mouse button. The evaluation for this system was made towards usability and ergonomics rather than performance of the gesture detection.

The detection of both types of gestures can be simplified to the detection of 2D gestures. The 3D gesture detection involves the reconstruction of the hand positions from a stereoscopic camera system, but the result of the 3D tracker is the same as the 2D tracker, e.g. the contour of the hand. Once the contour of the hand is acquired, finger detection is performed. The number of fingers in the hand and the position of the hands determine the gestures to be detected. That is, in the 2D hand gestures one hand is the ”pointer hand” and the other is the ”commanding hand”.

The performance of the gesture detection was not evaluated in this paper since it was evaluated in the previous work by the same team. Here, they focus on assessing the usability and ergonomics of system.

ARI [4]**.** ARI is an Augmented Reality Interface developed by OnTheGo Platforms. It enables android devices to recognize hand gestures through the device camera. They offer an SDK that, according to their website, detects natural and intuitive gestures without the need for extra hardware or

[4]https://www.otgplatforms.com/

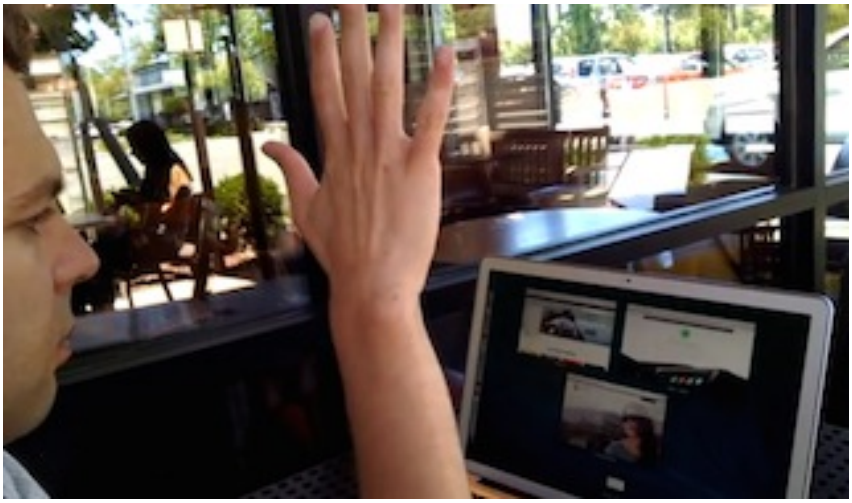


Figure 14: ARI. Image taken from the video available at their website. Here we can see the mission control in a Mac laptop being activated by the hand.

clothing. Their gesture vocabulary is not explicitly defined, but from videos available on the website the vocabulary goes from gesturing a square to take a picture, gesture a ”thumbs up” to share on social media, and controlling a robot with the hands. In figure 14 we can see the hand being used to activate mission control[5] in a Mac laptop. They recently integrated their solution with Metaio[6] and bring gesture detection into augmented reality. Since they are a private company, their gesture detection method is not described. But to the best of our knowledge, after using one of their sample applications, we believe they use background subtraction techniques to track the hand and possibly machine learning techniques to classify the gestures.

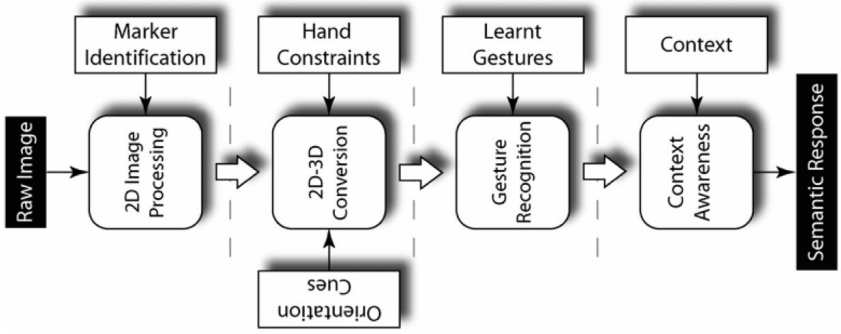


Figure 15: Dynamic Gesture Recognition system’s overview.

Dynamic Gesture Recognition (Joslin et al., 2005). This work presents the design of a context-aware dynamic gesture recognition system. The complexity of the gesture recognition is separated

[5]http://support.apple.com/kb/ht4689
[6]http://www.metaio.com/

in two areas. The first is identifying the gesture itself, and the second is understanding the motion compared to other specific or non-specific gestures.

Three constraints or considerations are presented as the core concepts of the design for the system. The first is that the system should work in *Generic Environments*, referring to different and cluttered backgrounds. The second is that the system does not provide a fixed application, in other words, the actions performed when detecting a gesture are not fixed because they depend on the context. This design aspect is still under research. The third is that the recognition of the gestures should be marker-less. This last constraint was not achieved and they use one marker at the center of the hand.

The system uses a module approach and has four components. The first component processes the image, the second component converts the image from 2D into 3D space, the third component recognizes the gestures using HMM, and the fourth component interprets the recognized gesture depending on the context. The last component is still under research. Although the context awareness component is not implemented, it is envisioned that the environmental context (quietness, light condition or geographic location) or the user context (age or disabilities) would influence the way a gesture is interpreted.

They do not present a formal evaluation of the system, but their internal evaluations show that their HMM is able to correctly recognize every gesture performed. The gestures used for this evaluation range form quotes signs to a grasp.

Figure 16: To the left we can see the wearable system for American Sign Language recognition. To the right we can see the camera view including the users nose.

Real-Time American Sign Language Recognition Using Desk and Wearable Computer Based Video (Starner et al., 1998). Two systems for recognition of American Sign Language (ASL) are described in this paper. Both systems use a single camera pointed at the hands of the user and HMM to recognize the gestures. The first system provides a full front view of the user from a desk mounted camera. For the second system a camera and a CPU unit are mounted on a cap worn by the user. A model of skin color is used to segment the hand. The image is scanned until a pixel matches the model, the pixel then is used as a seed to grow the region besides it. This segmentation approach results in real-time tracking of the hands without the aid of gloves or special markings. Although not implemented, an interesting workaround for when a skin color model is not known is presented for the second system. Since the camera mounted on the cap is looking down at the hand,

the user's nose is always in the picture and stays in the same place relative to it, as seen on figure 16. This would allow the system to use these pixels as a calibration object for the generation of a model of the skin color.

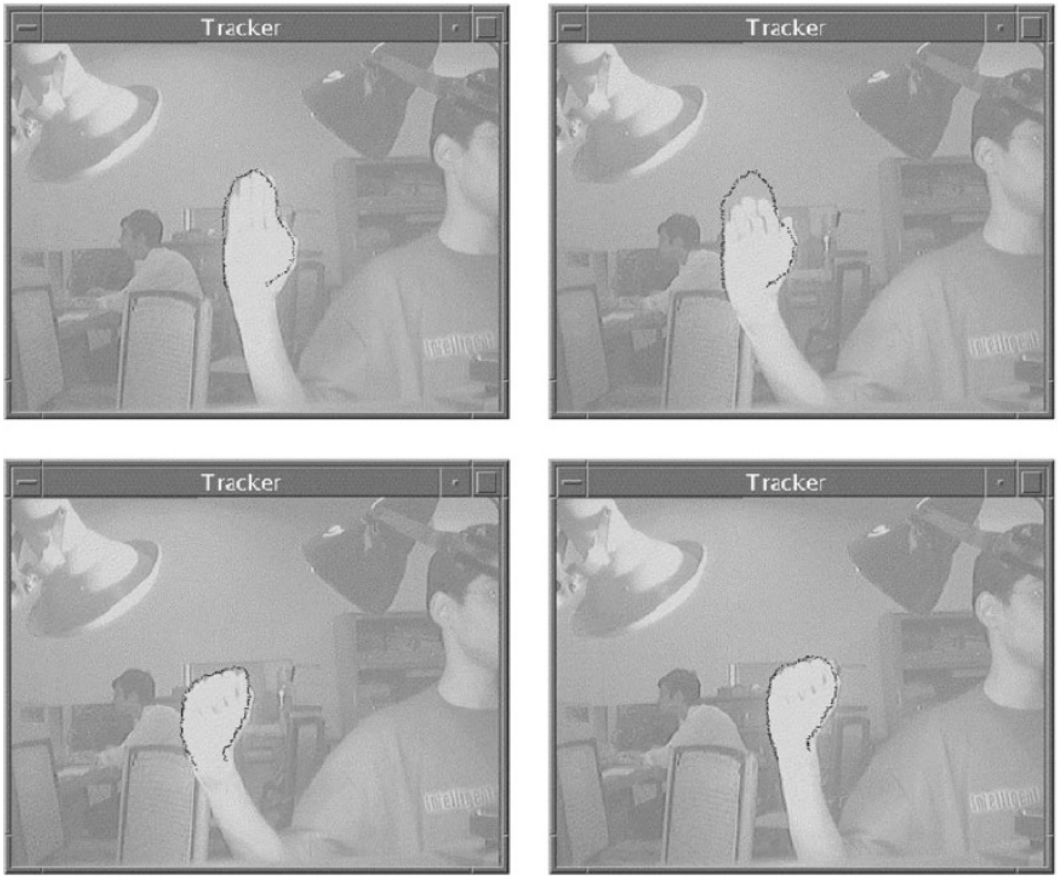


Figure 17: Recognition of dynamic hand gestures. Example of a gesture.

Recognition of dynamic hand gestures (Ramamoorthy et al., 2003). This work presents an engine for real time dynamic gesture recognition. The gestures are done with one hand and are defined as sequences of distinct hand shapes with a start and end state. The main contribution of this work is the method for detecting the gestures. The detection is based on HMM combining temporal and shape characteristics. It is robust to background clutter and does not require special gloves or markers. The detection is also able to adjust to changes in the shape of the hand while performing one of the gestures.

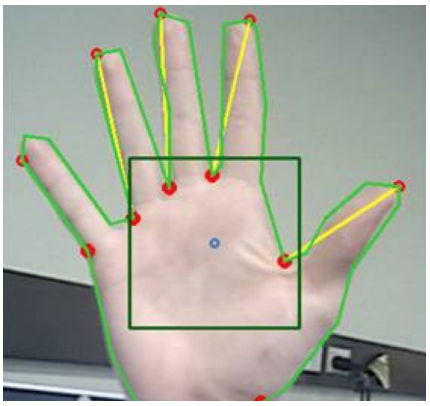
The basic algorithm for the gesture recognition is as follows. First the hand is detected in the image. Second, the tracker is initialized with the template of the detected hand shape. Third, a loop is started. While the hand is in the view it is tracked until a change in shape is detected. After a new shape is detected the tracker is initialized once more with the new detected template. Fourth, the gesture is recognized with HMM.

To demonstrate the engine in work a robot control system was developed and presented. In order

to control the robot the following five gestures are defined 1) Closed-open-forward 2) Closed-open-right 3) Closed-open-left 4) Open-closed-right and 5) Open-closed-left. These are dynamic gestures and are composed by two static gestures and a movement of the hand in between the static gestures. For example, the action for moving the robot forward is defined as a closed hand that moves towards the camera, stops, the hand changes shape to open, and moves away from the camera. They evaluate their HMMs, corresponding to the previous gestures, with five users. The data from the first two users was used as the training data, where each user performed the gesture 20 times. The recognition rate is presented for each user as seen on figure 18.

User	*C-O-F*	*C-O-R*	*C-O-L*	*O-C-R*	*O-C-L*
1	19/20	18/20	17/20	17/20	19/20
2	18/20	17/20	18/20	17/20	17/20
3	8/10	6/10	7/10	7/10	7/10
4	9/10	7/10	6/10	7/10	6/10
5	8/10	7/10	8/10	8/10	8/10

Figure 18: Recognition of dynamic hand gestures. Recognition rate results for their different gestures.



(a) Here we can see the two phases of the gesture detection. The number of held up fingers is determined by the defects (yellow lines) and the center of the hand moves outside the green box in a determined direction.

(b) Example of how the combination of gaze tracking and gesture detection can be applied. A robot is controlled by the user looking at it and giving commands with the hand.

Figure 19: Interacting with Objects in the Environment by Gaze and Hand Gestures.

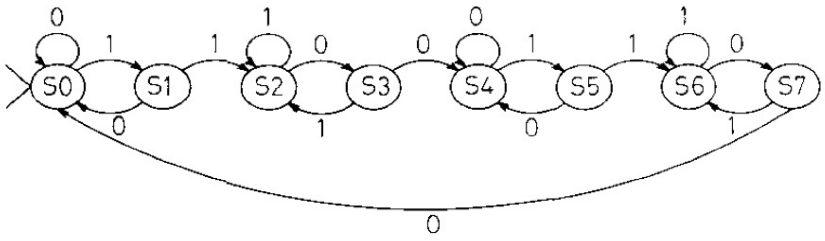
Interacting with Objects in the Environment by Gaze and Hand Gestures (Hales et al., 2013). In this work a method is presented that combines gaze tracking and gesture detection to interact with objects in the environment. The objects have visual markers attached to them so that when the user looks at the object the tag gets detected and selected from a database that maps the tags to the objects. Also, the hand is covered by a glove to make it easier to detect. After the selection of the object the user can interact with it by hand gestures. The gestures are defined as holding the hand up with lifted fingers and a movement of the hand in a particular direction. For example, the user holds one finger up and moves the hand towards the top of the screen. Since Gaze Tracking

is not within the scope of this thesis we are only addressing the gesture detection method from this project.

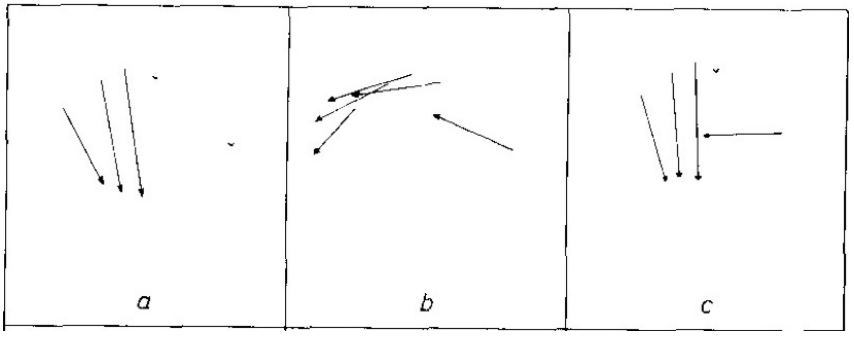
Four hardware components are used: 1) Gaze tracker glasses with two cameras that track one eye and capture the field of view of the subject respectively 2) A server in a remote PC receiving the video streams from both cameras 3) Visual markers attached to the objects and 4) The interaction objects or clients (e.g. robot or computer display).

The hand gesture recognition is done in the server application running on a remote PC. The gestures have two phases. The first phase consists of the static gestures with the fingers held up. The second phase consists of moving the static gesture towards a particular direction on screen. For the first phase the image is transformed to the HSV color space and filtered to detect the color of the glove. The resulting blobs are then filtered by size, leaving the biggest blob as the hand. The information about the shape of the blob is extracted and the convexity defects are determined. This defects are the ones that determine the number of fingers being held up. For the second phase the center of the hand is calculated and tracked. When the center moves outside a box placed around the hand it is classified as a movement and the gesture is detected.

Several proof-of-concept prototypes where developed using this technique. The most elaborated was implemented to control a robot. The robot could move in four directions: forward, backward, left, or right. For the first two the number of fingers determined the speed of the movement and for the latter two the fingers determined the number of turns to make.



(a) Here we can see the FSM implemented for the detection of gestures. It is used to guide the flow and recognition of gestures based on the characteristics of the hand's motion. States S0 and S1 represent the initial phase (phase I), states S2 and S3 represent motion to gesture (phase 2), states S4 and S5 represent the gesture recognition (phase 3), and states S6 and S1 represent motion to initial phase (phase 4)



(b) Examples of models of the fingertips movements for different gestures. a)left b)right c)up

Figure 20: Visual gesture recognition.

Visual gesture recognition (Davis and Shah, 1994). This work presents a method for hand gesture recognition based on computer vision. The gestures are represented by models of the trajectories of the fingertips. A marked glove is used to obtain the fingertips from the images. They present a gesture vocabulary with seven gestures: left, right, up, down, rotate, grab, and stop (figure 20b).

They obtain the models of the gesture by the motions of the fingertips. The fingertips move linearly so each trajectory can be approximated by a single vector. Each vector starts at the location of the fingertip before any motion is detected, and ends at the location of the gesture position.

This approach relies on four events, or phases, in the gestures. In other words, a gesture is represented by the following four phases 1) keep hand still (fixed) in start position until motion to gesture begins 2) Move fingers smoothly as hand moves to gesture position 3) Keep hand in gesture position for desired duration of gesture command 4) Move fingers smoothly as hand moves back to start position. These four phases occur in the same order, thus a FSM is used to guide the recognition of the gestures based on the trajectory of the fingertips (figure 20a). The image sequence is analyzed frame

Run	Frames	Left	Right	Up	Down	Rotate	Grab	Stop
1	200	✓	✓	✓	✓	✓	✓	✓
2	250	✓	✓	✓	✓	✓	✓	✓
3	250	✓	✓	✓	×	✓	✓	✓
4	250	✓	✓	✓	✓	✓	✓	✓
5	300	✓	✓	✓	✓	✓	✓	✓
6	300	✓	✓	✓	✓	✓	✓	✓
7	300	✓	✓	✓	✓	✓	✓	✓
8	300	✓	✓	✓	✓	✓	✓	✓
9	300	✓	✓	✓	✓	*	*	*
10	300	✓	✓	✓	✓	✓	✓	✓

✓ = recognised, × = not recognised, * = error in sequence

Figure 21: Visual gesture recognition. Results for their different gestures.

by frame to identify the 2D location of the fingertips and compute the trajectories using motion correspondence (mapping points in one image to points in the next image). The hand is considered to be moving when motion is present in more than three frames. Motion correspondence is used to map the starting points to the ending points, which can be used to determine the gesture. In order to recognize a gesture, they compare the unknown gesture with the models to determine if there is a match. They present overall results for ten sequences of images with almost perfects scores (figure 21).

Gesture modeling and recognition using finite state machines (Hong et al., 2000). Here, a state based approach to gesture learning and recognition is presented. Using spatial clustering and temporal alignment they built a FSM recognizer with temporal information. They define a gesture as an ordered sequence of states in the spatial-temporal space. The input to the gesture modeling and recognition is a simple feature set with 2D points representing the centers of the user's head and hand in the image sequence. The points are obtained by a real-time skin-color tracking method.

The FSM is built in two phases. First the number of states and their initial spatial parameters are calculated by dynamic k-means clustering on the training data. At this stage temporal information is not used. The training is done offline with several examples of each gesture performed by the user when requested. After the spatial information is learned the temporal information is added to the states. The recognition happens online at frame rate. Each feature vector is processed by all the gesture recognizers. Each gesture recognizer stays or jumps to the next state based on the spatial

Figure 22: Gesture modeling and recognition using finite state machines. Screenshot of the sample application "Simon Says". Here, we can see the tracking of the user's head and hand centers.

parameters an the time variable. When a recognizer reaches its final state, a gesture is recognized. If more than one recognizer reach their final state, the minimum average accumulated distance decides which one is recognized.

They do not explicitly present a gesture vocabulary, but they present an implementation of the game "Simon Says" (figure 22) as a proof-of-concept prototype where they support gestures such as left hand wave, right hand wave, and drawing a circle.

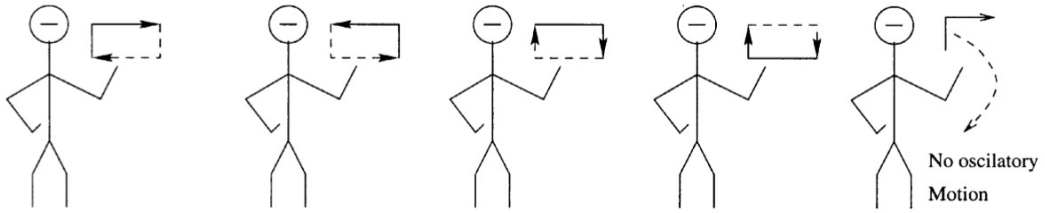


Figure 23: Visual understanding of dynamic hand gestures. Here, we can see the gesture vocabulary. The solid line indicated the start of the gesture along the direction of the arrow. The dashed line indicates the come-back phase of the gesture.

Visual understanding of dynamic hand gestures (Yeasin and Chaudhuri, 2000). In this work, a dynamic hand gesture recognition system is proposed to instruct a robot. Their gesture vocabulary includes the gestures come closer, go far, more right, move left, and emergency stop (figure 23). The gestures are defined based on the temporal signature of the hand motion. This signature is the input for a FSM recognizer. For example, they define the come closer gesture as one hand sweeping repeatedly, first towards the body and then away. They state that this typical motion pattern produces a unique temporal signature. Since they rely on this unique temporal signature, the relative motion of the hand (quick or slow) is not of importance for the signature.

In order to detect the gestures they define a set of deterministic FSMs to represent the temporal signatures presented in the gesture vocabulary. The gestures can be performed starting from any arbitrary spatial temporal position. All FSMs have the states start (S), up (U), down (D), left (L) and right (R). However, the gestures signatures can be represented only using two or three states. Three steps are done to detect a gesture.

a) Step 1: Capture the gesture data using a strategically placed camera.

b) Step 2: Extract the temporal signature of the gesture.

c) Step 3: Interpret the gesture using the production rule of the finite state machine model.

In step 2 they extract the temporal signature of a gesture by segmenting the data into sub-sequences that represent motions in only one direction (e.g. right or left). The motion analysis of the sub-sequences results in a signature of the gesture (e.g. R-L-L-R-L-R). In step 3 they decide which gesture the signature corresponds to. This is done by comparing the signature to the particular production rules, or grammar of the deterministic FSM, and declare a gesture.

For the evaluation they performed two experiments with motion estimation. The first experiment involves a synthetically generated sequence of images to detect the motion of an object. The second involves a natural sequence of images of natural hand motion (dynamic gestures). Both of their experiments show that the FSM always gives the correct interpretation of the motion.

2.4 Summary & Discussion

In this section we summarize and discuss the differences and similarities between the related work and our approach. We start with a summary of the types of gestures and control of the interaction. Following with the common set up of hardware and processing power. Then, we address the available results of the interaction and usability evaluations. Finally, we explain how our project is different to most of the discussed works. Figures 24 and 25 summarize the reviewed work.

2.4.1 Type of gestures and control of interaction

The characteristics of the hands in the gestures do not fully reflect the actions they represent affecting the intuitiveness of the system. The camera based systems usually define gestures based on the number of raised fingers. The systems that have depth sensing information define gestures based on the hands going towards or away from the camera which can be intuitive but people may not feel comfortable doing that in public. Others, like MIME (Colaço et al., 2013), define some gestures using two hands which gives the restriction to the user of having both hands free. Another common characteristic of these systems is the lack of control that the user has over the detection. Gestures that control the interaction or stop/cancel the current detection are not addressed. This takes control away from the user and may result in frustration. An example of a project that does include control of the interaction in their design is Ubi-Finger. They have a touch sensor to the left of the index finger (see figure 8) that serves as a initiator of the interaction. When the user clicks touches the sensor, the Ubi-Finger starts sensing for gestures. Another example of interaction control is the PTG (push-to-gesture) mechanism presented in AirTouch that allows the user to confirm a gesture.

2.4.2 Hardware and processing power

Most of the described systems are connected to desktop computers, use special servers, or include external processors that increase the processing power. This hardware setup allows for the systems

	Others	Us
Gestures & Control	No control over interaction and gestures represented by numbers, e.g gesturing a number one represents move forward.	More complex gestures that have a closer representation to their actions. Control over the interaction with gestures like *Init* and *End*.
Hardware and processing power.	Use of desktop computers to process information, IR cameras, 3D cameras, and wearable sensors.	2D Camera.
Evaluation.	Performance evaluation.	User experience and Interaction evaluation.

Table 1: Comparison between related work and our approach

to use complex techniques and algorithms, e.g. HMM, that allow the gesture detection to be more robust. Another common characteristic for the hardware selection is the use of special cameras. The use of infrared technology and depth cameras has been noticed as a popular selection for the type of camera. This, of course, increases the ability of the system to detect gestures in different light conditions and cluttered backgrounds, but it also decreases portability and availability to the public since extra hardware needs to be acquired.

2.4.3 Evaluation

The majority of the described systems evaluated their systems according to the performance of their gesture detection or classification approach. Four of them evaluated the system according to the usability and user experience.

In (Tsukada and Yasumura, 2002) their results include that 1) their gestures are easy to understand, 2) the selection of real-world devices by pointing with fingers is attractive, and 3) the selection of real-work devices by pointing is easy to understand.

In (Kim et al., 2014) they reach the conclusion that when a user is engaged in the activity being performed, they are willing to spend time defining gestures on their own. For example, a participant had experience in music and enjoyed spending time customizing gestures related to music.

In (Argyros and Lourakis, 2006) they present an interesting comparison between the 3D and 2D spaces. The former was preferred by novice users because they felt more comfortable gesturing in a 3D space with one hand. On the other hand, expert users preferred the latter because they find it faster to gesture in a 2D space with two hands.

In (Jalaliniya et al., 2013) the findings include that the combination of foot and hand gestures is an acceptable alternative to interact with images without direct contact.

2.4.4 How we are different

The first difference is the type of gestures. While some of the systems define intuitive gestures, we feel that we can support a more natural and intuitive connection between action and gesture.

The second difference is the type of hardware selection as well as the inclusion of extra hardware for the detection of gestures. As mentioned before, most of them involve big processing units

for the detection of gestures. We will only rely on the processing power and out of the box hardware of the selected eyewear computer for the gesture detection.

The third and main difference between us and the majority of the described work is that we evaluate interaction rather than performance. As mentioned in the previous subsection, only a small subset evaluate their systems in that way and we intend to take these works as guides for our evaluation.

Project	Recognition	Processing hardware	Gestures	Actions	Control of interaction	Evaluation
GestureWrist and GesturePad	Capacitive sensing	External signal-processing board	Combination of a hand shape and arm positions.	Examples: volume control of a worn MP3 player.	No	Proof-of-concept prototype
CompositeGesture	State Machine	Mobile devices	Defined by user. Motions captured by the sensors of the mobile device	Defined by user	No	Experimental study with interview regarding usability
GesturePendant	Neares Neighbor and HMMs	PC	Control gestures: Hand shape. User defined gestures: blob characteristics	Example control gestures: turn down the volume of the T.V. Example user defined gesture: turn on or off the T.V.	No	HMMs and Neares Neighbor performance
OmniTouch	Finger segmentation and flood fill	PC	Pointing	Actions similar to a mouse pointer, e.g. Click and/or drag.	No	User study evaluation: performance of the system
Skinput	SVM	PC	Finger taps in the body	Trigger events associated with the tapped location	No	User study evaluation: performance of the system
Ubi-Finger	Sensor motion detection	PC	Finger gestures	Control of home appliances. E.g. Turn on the T.V.	Yes	User study evaluation with questionnaire for subjective feedback
WUW	Simple Computer Vision based techniques	PC	Gestures with 2 hands	Zoom-in, Zoom-out, take picture, drawing in the air.	No	Proof-of-concept prototypes
AirTouch	HMM	Mobile device or wearable computer	Hand movements above the watch	Control of electronic devices, e.g. MP3 player.	Yes	User study evaluation: performance of the system
Touch-less Interaction with Medical Images using Hand & Foot gestures	Neural Networks	PC	Arm movements	Menu navigation and zoom-in and out pictures.	No	User study evaluation: performance of the system and usability

Figure 24: Summary of related work part 1

Mime	SVM	Wearable computer	Sngle handed gestures	depending on application. Examples: circle, point-and-click,	No	Proof-of-concept prototypes
Vision Based Interpretation of Hand Gestures	Hand characteristics (defects)	PC	Gestures with 1 or 2 hands	Mouse control	Yes	User study evaluation: usability and user experience
ARI	*	Wearable computer	Gestures with 1 or 2 hands	Diferent gestures depending on application. Examples: take picture, swipe, zoom in-out.	No	*
Dynamic Gesture Recognition	HMM	PC	1 or 2 hands with finger motions	Examples: quote signs and grasp.	No	Internal evaluation: recognition rate of HMMs
Recognition of dynamic hand gestures	HMM	PC	Dynamic gestures: 2 static gestures and a hand motion	Robot control: 1) forward 2) forward then right 3) forward then left 4) backward then right 5) backward then left	No	Recognition rate of HMMs. 2 users for trainning, 3 users for testing
Interacting with Objects in the Environment by Gaze and Hand Gestures	Hand characteristics (defects)	PC	Hold up the hand with lifted fingers and a movement in a particular direction	Robot control: 1) forward 2) backward 3) left 4) right	No	Proof-of-concept prototype
Visual gesture recognition	FSM	SPARC-1	Models of motions of the fingertips	1)left 2)right 3)up 4)down 5)grab 6)rotate 7)stop.	No	Overall results for image sequences
Gesture modeling and recognition using finite state machines	FSM	PC	Models of 2D points representing the centers of the user's head and hand in the image sequenc	1) Left hand wave 2) Right hand wave 3) Draw a circle 4) Draw a figure eight	No	Proof-of-concept prototype
Visual understanding of dynamic hand gestures	FSM	PC	Models of temporal signatures of the hand motion	Robot control: 1) come closer 2) go far 3) more right 4) move left 5)emergency stop	No	Two experiment. The first with synthetically generated sequence of images. The second involves a natural sequence of images of hand gestures

Figure 25: Summary of related work part 2. In the case of ARI, they are a private company and do not share their detection method or any evaluations of the system in their website.

3 Design

We followed an iterative design approach with four iterations for the development of a prototype that supports hand-based gestures as input modality for eyewear computers. The core of the prototype is the detection of hand-based gestures using a 2D camera. Here, we describe the steps, possibilities, and decisions throughout the iterations that make the gesture based interaction possible. In (Mitra and Acharya, 2007) a gesture is defined as meaningful and expressive body motions involving physical movements of the fingers, hands, arms, head, face, or body with the intent of: 1) conveying meaningful information or 2) interacting with the environment. They can be static (holding up a pose or configuration) or dynamic (with with pre-stroke, stroke, and post-stroke phases). Our final gesture vocabulary presents static and dynamic gestures that express meaningful information.

The rest of the chapter is as follows. Firstly, we describe the first iteration of the prototype. During this iteration an initial definition of the gestures emerged, rapid prototypes were built that led to the selection of the gesture detection method, and the final software and hardware choices were taken. Secondly, we describe the second iteration. During this iteration it was proven that the selected method could be implemented in eyewear computers and mobile devices. The iteration continues with the formal definition of the scenario. This led to an extension of the gesture vocabulary, the definition of requirements, and the architecture for the final prototype. Next, we address the third iteration. During this iteration, with the requirements from the second iteration, we implemented the first version of the prototype, performed the first formative evaluation, implemented the second version of the prototype, and finalize the iteration with the second formative evaluation of the prototype. Then, we follow with the fourth iteration. This iteration started with the analysis of the second formative evaluation results, implementation of the third version of the prototype, and finalized with the final evaluation of the system. Finally, we summarize this section.

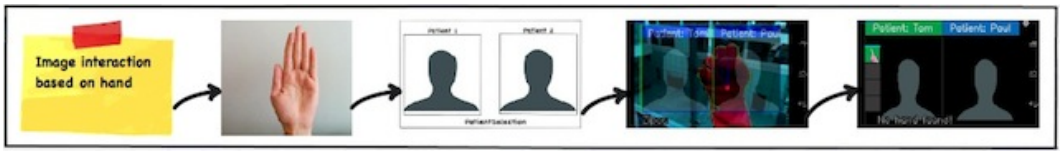


Figure 26: Summary of iterations. Here we can see some of the representative steps throughout the iterations.

3.1 First iteration - Defining gestures, detection method, and software & hardware

This thesis project started with the idea of supporting hand-based gesture interaction for eyewear computers in order to manipulate images. At the time, the image manipulation we envisioned was rotation and zooming of images, and switching through images in a carousel fashion. Also, we already had the idea that we needed a gesture to start or stop the interaction. One of the first challenges we face in order to define intuitive gestures is the connection they have to their actions. Mapping from concepts to gestures, or actions to gestures, can not always be straight forward (Mitra and Acharya, 2007). Not only the same gesture can have several meanings, but also the same meaning can be applied to different gestures. The iteration started with a brainstorming session where we gathered information about the type of gestures suitable for image interaction. The results of the session were not as expected, but it still gave us key points for the first definition of the gestures. With the gestures defined we were able to implement and evaluate gesture detection methods. Once

the method was selected, we concluded the iteration with the selection of hardware and software for the final prototype.

3.1.1 Brainstorming session

As previously mentioned, the iteration started with a brainstorming session. A detailed description of the session can be seen in Appendix A.

Brainstorming sessions require some stimuli like pictures or video to get the ideas flowing (Benyon, 2010). That is why for this activity the group was shown a picture in a board and was asked to perform actions with the picture first by touching it, and then by performing gestures in the air. The actions to perform on the images were the ones we envisioned at the time, e.g. rotation, zooming, switching images in a carousel fashion, and starting or stopping the interaction. When a participant made a suggestion or performed actions on the images we encouraged the rest of the participants to discuss them. This discussion helped in our gestures definition because we could gather information about what the people would be comfortable with when interacting with a system like the one we developed. It also gave us extra information and even guidelines on how to design the gestures, e.g. the start or stop gesture should be something that one does not normally do.

The desired outcome for the session was to find a set of gestures for the user to interact with the images. The results are summarized in table 2. Although we were not able to have a concrete

Action	Suggested Gestures
Rotate	• Place hand (5 fingers) in front of face/camera and rotate to desired direction • Place 3 fingers pointing towards himself and rotate to desired direction • Place 2 fingers (L-shaped) and move them to desired direction • Place 1 finger towards user and make a rotate gesture (inverted U) towards desired direction
Zoom	• Pinch gesture with 2 fingers closing or opening the gap in between • Pinch gesture with 2 fingers using both hands and bringing hands closer or apart
Swipe	• Swipe movement using 1 or two fingers to the sides • Swipe movement using 1 finger up/down
Start or Stop	• Open hand with the palm towards user • ”Splash” hand towards the user

Table 2: Results of brainstorming session for the envisioned gestures at the time. These are the most popular suggestions for each gesture.

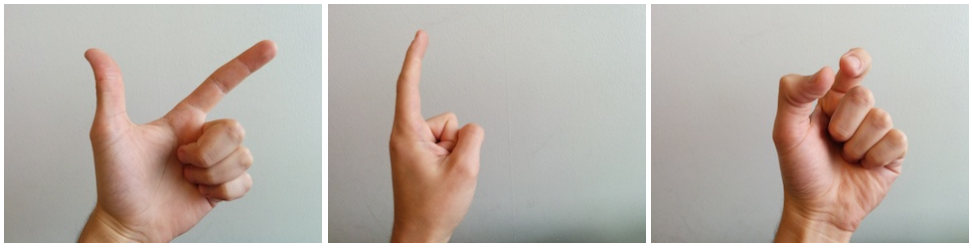
definition of the gestures at the end of the session, we did gather important key points from the comments and general thoughts of the participants that helped with the definition of natural and intuitive gestures for the envisioned actions. These key points are 1) that the gestures should be

performed with the palm of the hand facing the user and 2) that the start or stop gesture should be something that one would not normally do.

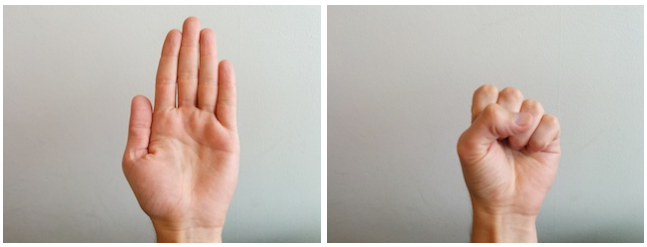
3.1.2 First definition of gestures

Argyros and Lourakis (2006) present the following criteria for the definition of gestures. First, they state that a gesture should be easy to learn and use, or *intuitive and ergonomic*. Second, they state that a gesture should facilitate automatic interpretation for its use, or *lack of ambiguity and ease of recognition*. With the information gathered in the brainstorming session and the previous criteria we did our first definition of the gestures.

The first definition of gestures included the actions Rotate, Swipe, Zoom, Init, and End. Following the previously mentioned criteria and the results of the brainstorming session we defined the gestures as follows.



(a) Rotate pre-stroke. (b) Swipe pre-stroke. (c) Zoom pre-stroke.



(d) Init static gesture. (e) End static gesture.

Figure 27: First gesture definition.

a) **Rotate.** Thumb and index finger in an L-shape with the rest of the fingers tucked in. This gesture is dynamic and would rotate the picture displayed on screen. The pre-stroke is seen in figure 27a, the stroke is the hand being tilted left or right, and the post-stroke is the hand titled towards the desired rotation direction.

b) **Swipe.** Index finger lifted with the rest of the fingers tucked in. This gesture is dynamic and would swipe through the pictures displayed on screen in a carousel fashion. The pre-stroke is seen in figure 27b, the stroke is moving the hand to the left or right, and the post-stroke is the hand at the left or right of the initial position.

c) **Zoom.** Thumb and index finger in a pinch shape with the rest of the fingers tucked in. This gesture is dynamic and would zoom in and out the picture displayed on screen. The pre-

stroke is seen in figure 27c, the stroke is the fingers moving closer together or farther apart, the post-stroke is the fingers with a smaller or bigger distance than the initial distance.

d) **Init.** All fingers together and fully extended. This gesture is static and would start the interaction with the system.

e) **End.** All fingers together and fully flexed. This gesture is static and would stop the interaction with the system.

Regarding the stop of the detection, the suggestions from the participants were that the gesture should be the same as the start of start of the interaction. After some consideration we decided not to follow the suggestions in this particular gesture and define it following the *lack of ambiguity* and *intuitive* criteria, thus we define the gesture by inverse logic. In other words, if the fully extended hand means start, the fully closed hand should mean stop.

3.1.3 Gesture detection method

Once we had the gestures definition, we proceeded with the development of rapid prototypes in order to select a gesture detection method that best fitted the type of gestures. For these developments we used Python and OpenCV [7]. The latter is one of the most popular, if not the most, computer vision libraries. The use of computer vision for gesture detection is addressed in detail in chapter 4. The only special reason we chose this combination of software is because we had been working with them not more than two weeks prior to the start of this project. We needed to develop quick prototypes and using other libraries and languages would have taken more time.

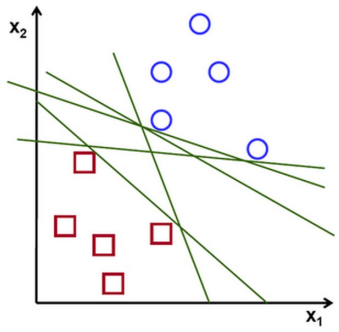


Figure 28: Example of the SVM classifier. Given a set of 2D points it finds an straight line separating both classes.

With these rapid prototypes our main goal was to find 1) a reliable and fast classification method, or 2) a reliable and fast hand segmentation method. In order to accomplish the previous goal we implemented several algorithms and techniques. Some were quickly discarded for being slow and unreliable, but here are two that are worth mentioning. Support Vector Machine, or SVM[8], is a discriminative classifier defined by a separating hyperplane. This approach is used by Skinput (Harrison et al., 2010) and Mime (Colaço et al., 2013). It takes labeled training data and the algorithm defines a hyperplane that separates the data into categories by which new data is also categorized. Another method used in these rapid prototypes was K-Means Clustering[9]. The K-Means method

[7] http://opencv.org
[8] http://docs.opencv.org/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html
[9] http://docs.opencv.org/modules/ml/doc/k_nearest_neighbors.html

clusters the given data according to the desired feature, color in our case, in k groups or clusters. Both are very good approaches for classification and segmentation, respectively. The downside to these approaches, as well as the rest, is that they require a lot of processing power. Even though they where developed and tested in a laptop, their performance was slow when processing a video. Also, the training phase of the machine learning algorithms takes time and in most of the cases requires very heavy computations. We already had in mind that the final prototype was going to be implemented for smart glasses so these methods were not suitable for us.

One of the rapid prototypes we implemented was a clone of the method proposed by Hales et al. (2013). This method is simple, light, and showed a good performance. It relies on the characteristics of the segmented object, e.g. the hand, and it is possible to track the hand frame by frame. Thus, we decided to take that work as a guide and implement a similar method. Our implementation of the gesture detection method is fully discussed in the next chapter.

3.1.4 Software & hardware

With the gesture detection method selected, the next step was to select the software and hardware for the final prototype. There are many options for computer vision libraries that support our needs. A simple search in Google throws libraries like the Point Cloud Library[10], the simpleCV framework[11], and the VXL libraries[12]. The are also many options for eyewear computers, with Google Glass[13] being the most popular.

Android and OpenCV. The software decision was made almost immediately after selecting the gesture detection method. We had a strong desire to work with the Android SDK[14] and OpenCV. As previously mentioned, experience developing with this library was acquired almost at the same period this project started but with the Python language. Studying the documentation of the OpenCV libraries for Android, OpenCV4Android[15], we concluded that the previous experience obtained for Python could be also applied for Android and continue to learn about this computer vision library.

Figure 29: Vuzix M100 Smart Glasses.

[10] http://pointclouds.org/
[11] http://simplecv.org/
[12] http://vxl.sourceforge.net/
[13] http://www.google.com/glass/start/
[14] http://developer.android.com/sdk/index.html
[15] http://opencv.org/platforms/android.html

Vuzix M100 Smart Glasses. Two sets of smart glasses where available to us: Google Glass and Vuzix M100 Smart Glasses[16]. Both run on Android and can support OpenCV. Once we had our Software selected, we implemented a "Hello World!" type application as our version 0 of the prototype. This was a very simple application supporting the detection of a hand frame by frame, which helped us to compare both smart glasses. We conducted informal and internal testings to pick the final eyewear computer and focus the implementation on this device. The final selection was the Vuzix M100 Smart Glasses. The main reasons for this choice are 1) performance and 2) application deployment. When comparing the performance of the mock-up application, the M100 showed dominance over Google Glass. Also, it is much easier to deploy an application in the Vuzix M100 than in Google Glass. According to the online documentation[17], to be able to deploy an application, or *Glassware* as they call it, directly on Glass the developer needs to use the GDK (Glass Development Kit) which would decrease the portability of our prototype. It is not our goal to develop a prototype that is portable between eyewear computers, but internally we had the need to be able to test on several devices, specially mobile devices, since the smart glasses where not available to us on demand. The Vuzix M100 fit better our internal requirements.

3.2 Second iteration - Scenario definition, requirements, and architecture

Once all the decisions in the previous iteration were done, we started the second iteration with the implementation of the gesture detection method for the Vuzix M100. During the time of the implementation, we knew we had to project our idea into a suitable use case scenario. Shahram Jalaliniya, one of our supervisors, referred us to the ward rounds of surgeons and shared his results of a yet unpublished ethnographical study in a Danish Hospital where he shadowed, and recorded, the activities of a surgeon during ward rounds. We complemented Shahram's work with the results of Bardram and Bossen (2005) and Carlsson et al. (2006), and defined the scenario presented in chapter 1 (see page 6). The iteration continued with an analysis of the scenario, which resulted in 1) the definition of the requirements for the final prototype, 2) the first redefinition of the gestures, and 3) the definition of the system architecture.

3.2.1 Requirements

The scenario presented in chapter 1 (see page 6) is an abstraction of the activities performed by a surgeon during a regular ward round. After an analysis of the scenario, we detected repetitive steps (letters A-B), as well as activities (numbers 1-6) specific to each step. We take these steps and activities as the starting point for the design of our prototype. The result of the analysis is the definition of our requirements.

1. The interaction with the system must be touch-less and using hand-based gestures.
2. The final definition of gestures must be intuitive, ergonomic, with no ambiguity, and easy to recognize.
3. The system must provide the user control of the interaction with special hand gestures.
4. The system must support three states representing the following steps in a ward round: patient selection, record viewing, and image interaction. These will be called overall states.
5. The system must support gestures that map to the specific activities of each state.

[16] http://www.vuzix.com/consumer/products_m100/
[17] http://www.google.com/glass/start/

(a) Patient Selection: select a patient.

(b) Record Viewing: browse the patient's available x-rays and select an x-ray.

(c) Image Interaction: rotating and zooming the image.

6. The system must be implemented for the Vuzix M100 Smart Glasses.

After defining the requirements, we started to adapt the gestures definition from the first iteration. It was clear that we needed to add a gesture for selection. Also, the implementation of the gesture detection method resulted in the redefinition of other gestures.

3.2.2 Second definition of gestures

The implementation of the gesture detection method and the definition of requirements resulted in changes to our first gesture definition. The former gave us the possibility to adapt and optimize the gestures definition and the detection method. The latter shaped the final vocabulary of gestures. Here we describe the changes to the gestures definitions as well as the inclusion of a new gesture.

As previously mentioned, the selected gesture detection method uses the characteristics of the segmented blob, e.g. the hand, in order to classify the gestures. This means that we needed to define the characteristics that best describe, and at the same time differentiate, the gestures. These characteristics are called convexity defects (figure 46) that represent the fingers being lifted. We describe them in detail in the next chapter.

(a) PointSelect pre-stroke. (b) Zoom pre-stroke. (c) Swipe pre-stroke.

Figure 30: Second gesture definition. Here, we show the addition of PointSelect, and changes in Zoom and Swipe. Rotate, Init, and End remained the same.

a) **PointSelect.** Index finger lifted with the rest of the fingers tucked in. This gesture is dynamic and was added in this iteration. The tip of the finger represents the pointer location to select objects displayed on screen. Its pre-stroke can be seen in figure 30a, the stroke is the finger moving around the screen changing the pointer location, and the post-stroke is lifting a second finger.

   It is natural and intuitive to point at something with the index finger, thus the decision for the pre-stroke was easy. On the other hand, the pose to end the gesture presented more difficulties. This post-stroke needed to be something to confirm that the pointing position wants to be selected, e.g. click that position. We wanted to maintain the characteristics of the pre-stroke, but with an extra element that made it different enough for it to be considered a different gesture. We consider the inclusion of an extra finger to the pre-stroke is that extra element.

b) **Zoom.** Index and middle fingers lifted with the rest of the fingers tucked in. This gesture is modified in this iteration and would zoom in and out the picture displayed on screen. The pre-stroke is seen in figure 30b, the stroke is the fingers moving closer together or farther apart, and the post-stroke is the fingers with a smaller or bigger distance than the initial distance.
The main motivation for the redefinition of this gesture was that it was difficult for us to define the characteristics of the pre-stroke. Every suggestion in the brainstorming session for this gesture involved the "pinch-and-zoom" motion. Thus, for the second definition we wanted to keep the same finger motions as the first definition, e.g. bring fingers together or farther apart, but with explicit characteristics of the convexity defects. We consider that using the index and middle fingers to represent the desired finger motions is the best option.

c) **Swipe.** Index and middle fingers lifted with the rest of the fingers tucked in. This gesture is modified in this iteration and would swipe through the pictures displayed on screen in a carousel fashion. The pre-stroke is seen in figure 30c, the stroke is moving the hand to the left or right, and the post-stroke is the hand towards the left or right of the initial position.

   The inclusion and definition of *PointSelect* resulted in the need to modify this gesture. With the selected method it would not be possible to support two gestures with the same pre-stroke *in the same overall state*. Looking back at the results of the brainstorming session we find that the participants also suggested the gesture to be defined as "2 fingers moving to the side". Thus, we modified the definition to fit this suggestion.

We define the pre-stroke for *Swipe* and *Zoom* the same way because they are not used in the same state, and their stroke and post-stroke are different. In retrospective, we acknowledge this definition could have confused the users. This is discussed in detail in chapter 6.

3.2.3 Architecture

At the beginning of this chapter we state that the core of the prototype is the detection of hand-based gestures using a 2D camera. In order to detect the gestures in the image acquired from the camera we need Computer Vision, one of the fields inside Image Processing. Computer Vision is used to understand what is seen in an image through algorithms and computer programs (Solem, 2012). A general framework to work with computer vision is described by Moeslund (2012). We discuss in detail how we implement the framework in the next chapter. The steps of this framework are illustrated in figure 32 with the orange marks. Each step represents a block and they are explained next.

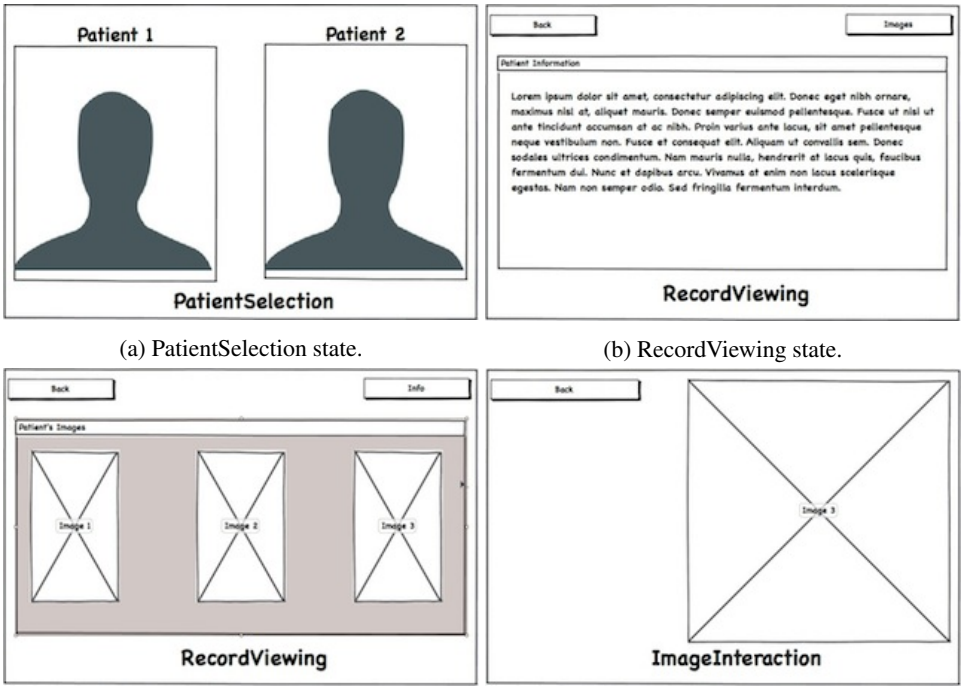
a) **Image aquisition.** Here, the setup before acquiring pictures is done. From choosing the type of the camera to preparing the lightning conditions.

b) **Pre-processing.** Once the image is obtained, operations are applied to the it before the actual processing begins. These operations can go from changing the color space to cropping the image.

c) **Segmentation.** Here, the objects in the image are segmented and important information is extracted, for example the size of the foreground object.

d) **Blob analysis.** Here, the segmented objects are extracted and classified.

We apply the work flow of this framework in our architecture because it fits perfectly with our detection method. We implement a modular architecture (figure 32) with several components which are explained next.



(a) PatientSelection state. (b) RecordViewing state.

(c) RecordViewing state. (d) ImageInteraction state.

Figure 31: Mock-ups of the prototype.

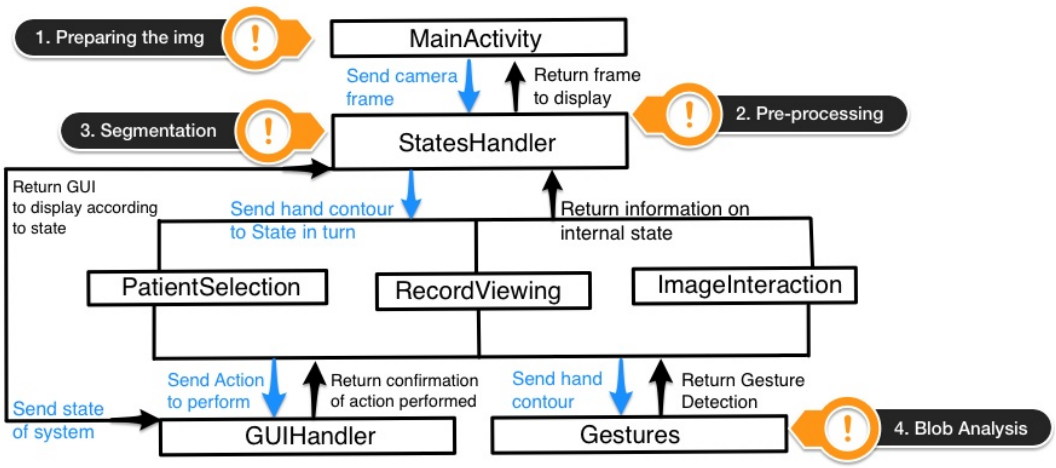


Figure 32: Overview of the Architecture including the Image processing framework.

a) **'Main Activity'** handles the communication with the camera and the display. It receives the frames, sets the parameters of the camera before the next frame is received, and returns the frame to display.

b) **'StatesHandler'** can be seen as an orchestrator since it controls the overall states of the system. The preprocessing and segmentation of the image is performed in this component. The result of the segmentation is sent to the state in turn. After the frame is completely processed, it performs the appropriate changes to decide the next overall state, e.g. if the patient has been selected move to the record viewing state. Finally, it returns the frame to display to **Main Activity**.

c) **'PatientSelection'**, **'RecordViewing'**, and **'ImageInteraction'** are the repetitive steps of the surgeon in our ward round scenario (see page 6). These components define the overall states of the system. The component in turn, depending on the state, decides which gestures are detected.

d) **'Gestures'** analyses the received blob, e.g. segmented hand, and detects the gestures performed by the user.

e) **'GUIHandler'** handles what happens on the user interface according to the overall state and the gesture being performed by the user.

3.2.4 Workflows

In this section we dig deeper into the overall states. The workflow in our design resembles an FSM approach like the ones presented by Davis and Shah (1994), Hong et al. (2000), and Yeasin and Chaudhuri (2000). In our approach, the **'Gestures'** component acts like the classifier but it does not decide when to change states. **'StatesHandler'** is the one that decides when to change states.

Each overall states possesses internal states essential for the control of the interaction. In this subsection we describe how the system moves between states. We first address the *overall states*, which dictate the gestures to be detected. We then follow with the *internal states*, which describe the behavior between gestures.

Overall states. The overall states represent sequential repetitive activities from the surgeon in our ward round scenario (see page 6). Each overall state has actions specific to it which are mapped to dynamic gestures. As seen on table 3, having actions specific to overall states results in gestures specific to overall states. This reduces the number of gestures supported in each state. This approach is similar to the one presented by Davis and Shah (1994). Where they reduce the search complexity by comparing to models where the comparison is logical (Davis and Shah, 1994).

All overall states support the *Init* and *End* static gestures. In the previous section we placed these overall states in our architecture and here we describe them in more detail. The workflow between overall states can be seen in figure 33, and mock-ups of the graphic user interface are shown in figure 31.

Figure 33: Overview of the Overall states of the system and the actions resulting in a change of state.

PatientSelection	RecordViewing	ImageInteraction
• Init • End • PointSelect	• Init • End • PointSelect • Swipe	• Init • End • PointSelect • Rotate • Zoom

Table 3: Overview of the Overall States and the gestures supported on each state.

a) **Patient selection.** The system starts in this state. Here, the surgeon has to select, with the *PointSelect* gesture, between the available patients displayed on the screen. The available dynamic gesture in this state is *PointSelect*.

b) **Record viewing.** When a patient is selected the system moves to this state. Here, the latest information on the treatment is available. The surgeon can *PointSelect* the Images button to see the available x-rays of the patient. Then, the surgeon can *Swipe* through the x-rays and *PointSelect* one. The available dynamic gestures in this state are *PointSelect*, and *Swipe*.

c) **Image interaction.** When an x-ray is clicked the system moves to the this state. Here, the x-ray is displayed in a bigger scale and the surgeon can *Rotate* and *Zoom* the image. The available dynamic gestures in this state are *PointSelect*, *Rotate*, and *Zoom*.

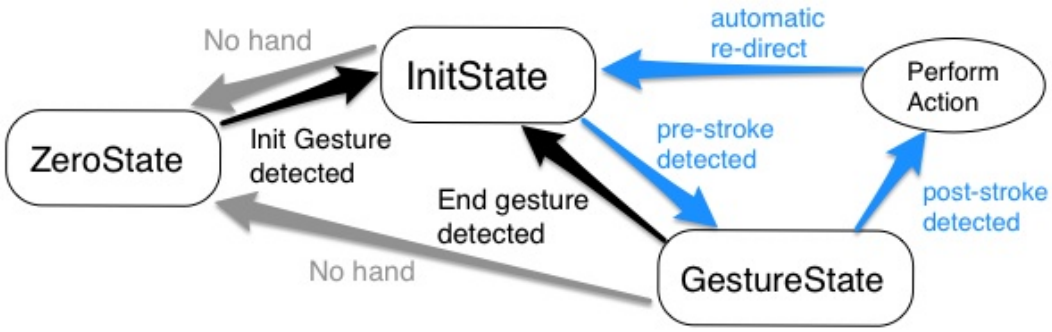


Figure 34: Generalized workflow of the internal states. Each available gesture in the overall states represent an internal state. If no hand is detected in screen for 40 seconds the system automatically goes to *ZeroState* in **PatientSelection**.

Internal states. The internal states are important for the control of the interaction. Here is where the static gestures *Init* and *End* play a central role. A generalized workflow these internal states can be seen in figure 34. Each dynamic gesture supported in an overall state represent an internal state. Thus, every overall estate has at least three internal states: **ZeroState**, **InitState**, and **PointSelectState**. The overall state in turn always starts in **InitState**. From this state, when a pre-stroke is detected the system moves to the **GestureState**, e.g. if the *Zoom* pre-stroke was detected the system moves to **ZoomState**. While the system is in the **GestureState** it means that no gesture will be searched in the image other than the gesture's post-stroke, e.g. the system will only look

for *Zoom*'s post-stroke. From here, the system can either detect the post-stroke for the gesture in turn or the *End* gesture. The former would cause an action to be performed, e.g. the post-stroke for *PointSelect* was detected and the click is processed. The latter would cause the system to return to **InitState**.

It is important to mention a special definition in the workflow of *PointSelect*. When the internal state is **PointSelectState**, the system allows a loop to stay in the same internal state if *PointSelect*'s pre-stroke is detected again. This allows the user to keep pointing at different locations until *PointSelect*'s post-stroke or *End* is detected.

The **ZeroState** means that no hand has been detected for 40 seconds. When this happens the system automatically assumes the interaction is over and goes to *ZeroState* in **PatientSelection**. If the system is in *ZeroState* the user then has to restart the interaction with the *Init* gesture.

3.3 Third iteration - Prototype versions 1 & 2, and formative evaluations

At this point in the development process we had a scenario for the prototype, a gesture vocabulary, and the architecture for the prototype. The third iteration began with the incorporation of the scenario with the gesture detection method, e.g. the first version of the prototype. Due to internal testings and the first formative evaluation, the prototype went through several changes and improvements, which we separate in four categories 1) addition to workflow 2) third definition of gestures 3) selection of objects and 4) feedback. The iteration ends with the second formative evaluation. In the rest of the section we address each category of the changes.

3.3.1 Addition to workflow

A problem in the workflow of the internal states was found during the internal evaluations. Before the adaptation of the scenario in the prototype, the internal testings focused on particular strokes of the gestures being detected. With the first version of the prototype ready, the tests were more fluid and with a different focus. This led to the realization that when a post-stroke was detected there was not enough time to change the hand pose, or even take the hand out of the screen, resulting in a new gesture detection. Thus, we decided to include a waiting time of two seconds when any post-stroke is detected. Also, we extended this idea and defined a waiting time of one seconds when any pre-stroke is detected. With this inclusion of a waiting time the prototype can be used in a more relaxed way.

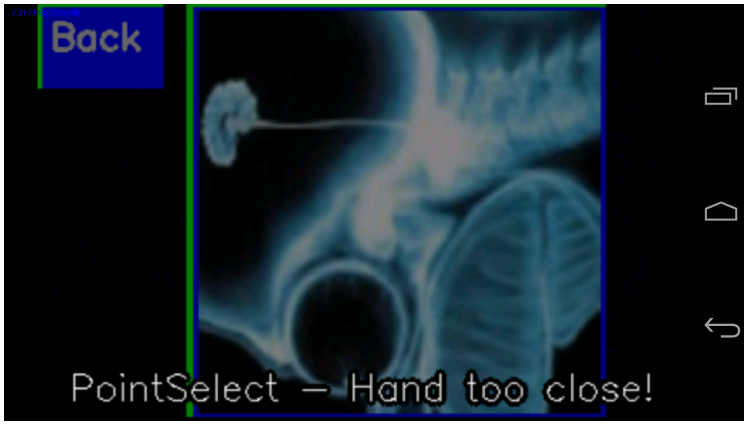
3.3.2 Third definition of gestures

During the implementation phase we detected a lot of false negative classifications of the *Init* gesture. After carefully analyzing the situation, we came to the conclusion that the defects of *Init* and *End* were too similar. During the first iteration we defined *Init* based on the suggestions from the participants of the brainstorming session, and *End* was defined as *Init*'s inverse. We wanted to maintain the relation for this new definition, and came to the conclusion that modifying *End* would result in a unintuitive gesture. Thus, we decided to change *Init*'s definition. Figure 35 shows the new definition. Another change during this iteration was the scope of *PointSelect*. We decided that the user must be able to navigate a zoomed-in image. The rest of the gestures remained unchanged.

a) **Init.** All fingers extended and separated from each other. This gesture is static and it is used to start the interaction with the system. Changing the fingers from together to separated was enough to get rid of the negative classifications. Also, this definition still follows the suggestions from the brainstorming session.

Figure 35: Redefinition of the *Init* gesture.

b) **PointSelect.** The definition of this gesture remained the same, but its scope changed. When the system is displaying a zoomed image, the user can *PointSelect* a new area of interest (figure 36). This causes the new selected point to go to the center of the displayed image.



(a) Imaged rotated and zoomed in.

(b) Different center of focus on image.

Figure 36: Example of a new center of focus in a zoomed in image.

3.3.3 Selection of objects

This improvement involved the selection of objects in the screen. Because of our definition of *PointSelect* and the constraints in our gesture detection method, e.g. size of the contour, it was very difficult for the user to select anything below the middle of the screen, meaning that almost 50% of the screen was un-clickable. When the user tried to select something in the lower part of the screen the hand contour was very small and it would go undetected. The users needed to place the hand in very uncomfortable and awkward positions to select objects in this area violating the *ergonomics* principle. And when it was big enough, it was so big that the lowest clickable point was a bit lower than the middle of the screen.

In order to make the whole screen clickable and maintain the *ergonomics* principle we decided a good solution was to map the upper half of the screen to the whole screen. In order to achieve this mapping we needed to block the live feed from the camera, e.g. display the background in black. Since the hand of the user would not be visible anymore we faced a new question: What sort of feedback of the hand would the user prefer? In order to answer this question we designed our first formative evaluation. The full description of the evaluation can be seen in appendix B. Three settings were presented to the participants. Each setting had the same task of clicking the rectangles displayed in the screen with *PointSelect*.

a) **Setting A.** The first setting shows the hand contour in green with the live feed from the camera.

b) **Setting B.** The second setting shows the hand contour in green without the live feed from the camera.

c) **Setting C.** The third setting shows dots placed on the tip of the fingers without the live feed from the camera.

Even though our solution for the un-clickable lower part of the screen was to block the camera feed and display the background in black (settings B and C), we needed to make sure that this setting was preferred over the initial set up (setting A).

The participants chose setting B (figure 37) as the easiest, more convenient, and accurate when clicking. With this new setting we were able to map the upper part of the screen to the entire screen and make it 100% clickable with a simple transfer function. The mapping of the screen is described in the next chapter.

3.3.4 Feedback

This category of improvements involve the feedback given to the user regarding the gestures. The first inclusion in this category was the ability to hover the buttons. Before this modification there was no way of knowing which objects were clickable. Now, the green rectangle surrounding the buttons turns red letting the user know that a button is being hovered.

The second inclusion in this category was to display graphics when a pre-stroke is detected. These graphics include arrows describing the stroke of the gesture. Before, only the *pointSelect* gesture gave feedback to the user by showing a purple dot as the pointer. Now, whenever a pre-stroke is detected, a graphic is shown with arrows to remind the user of the stroke motion of the gesture. The result of the third iteration can be seen in figure 38.

After implementing the previously described changes, we finalize the iteration with the second formative evaluation of the system. This evaluation resulted in even further changes to the prototype, e.g. iteration four, which are address in the next section.

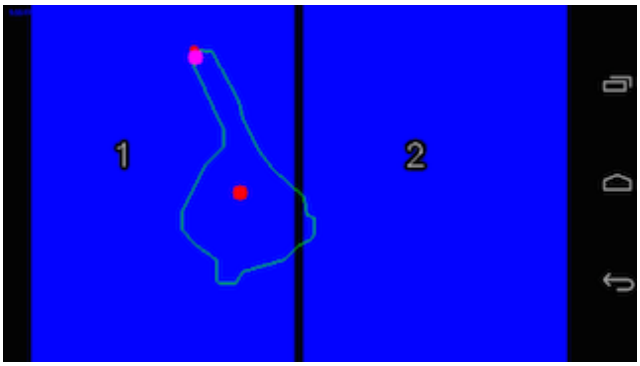


Figure 37: Result of the formative evaluation for choosing the hand feedback. Here we show setting B, where the hand contour is presented to the user in green and the live feed from the camera is blocked. This allowed us to map the upper half of the screen to the whole screen.

3.4 Fourth iteration - Prototype version 3 and final evaluation

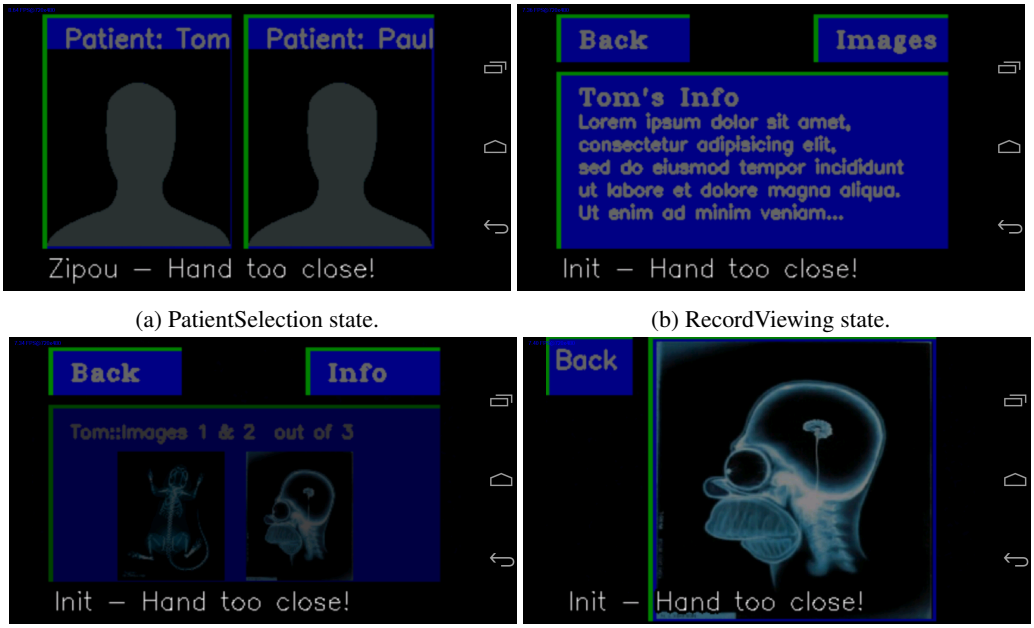
The fourth iteration started with the analysis of the results from the second formative evaluation of the system. The evaluation is described in detail in Appendix C. The analysis of the results pointed out that the user feedback was not enough. Thus, further changes in the prototype were implemented resulting in the third version of the prototype. Finally, the iteration ended with the final evaluation of the prototype, described in detail in chapter 5. The rest of the section is as follows. We begin with a short description of the second formative evaluation and present its results. We follow with an analysis of these results, and then present the changes to the prototype in this iteration.

3.4.1 Second formative evaluation

The second formative evaluation of the system was designed to take place in two phases. The goal of the first phase was to evaluate the usability of the system and the user experience. The second phase had the goal of measuring several aspects of *PointSelect*'s performance. Here, we address the former. During the first phase the participants had to follow a scenario, similar to the one presented in chapter 1 (see page 6), that contained all the dynamic gestures defined in our vocabulary. After the scenario, they were asked to answer a questionnaire with 3 parts.

Results. In the first part of the questionnaire, the participants were asked to identify the gestures. Figure 39a shows the results for this part, the optimal result is 6. We can see that one participant had problems differentiating between *Zoom* and *Swipe*, choosing the former twice and not choosing the latter at all. The second part of the questionnaire included the questions 1)How Intuitive are the Gestures? 2)How easy is to perform the Gestures? and 3)How easy are the Gestures to remember?. The answer to this questions was in the likert scale. We can see in figure 39c the lowest scoring gestures are *Zoom* and *Swipe*. The third part includes the question How responsive do you find the system?, also in the likert scale. Figure 39b shows that the participants give a low score to the responsiveness of the system.

Analysis. We considered the results for part 1 and 2 of the questionnaire to be acceptable. On the other hand, the result in part 3 told us that the responsiveness of the system needed to be improved. At the time we thought that improving the feedback would reflect positively on all scores, not only responsiveness. Thus, we decided to focus all our attention on the user feedback for the fourth



(a) PatientSelection state.

(b) RecordViewing state.

(c) RecordViewing showing the patient's x-rays.

(d) ImageInteraction state.

Figure 38: Second version of the prototype.

iteration. In order to improve the feedback of the system we decided the best way to know what the user needed was to go through the videos of the evaluation sessions. Analyzing the videos we found the following problems.

a) **Wrong feedback.** We found that we were giving the wrong kind of feedback to the user. Specifically, we were expecting the user to read the feedback. Also, because the feedback given was in text, we had to place it in odd positions in the screen, e.g. at the bottom, where the user was not expecting it. This caused a lot of confusion in the participants.

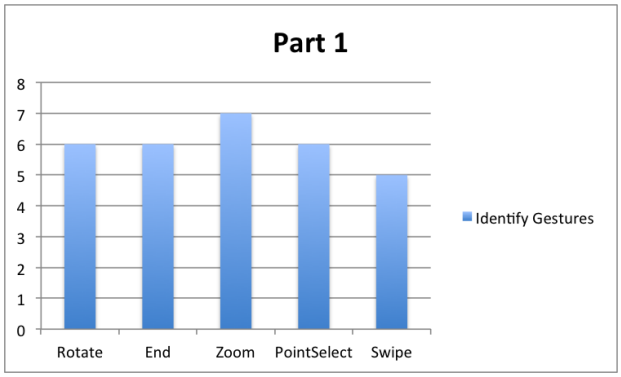
b) **Lack of feedback.** We found that we were not giving feedback on important aspects about the states and gestures, e.g. what gestures are available at the moment.

c) **Time for pointing.** Although most of the participants did not suggest or complained about the size of the objects (only 1 participant complained), we noticed that some time was being lost in moving the hand across the screen in order to select the buttons. Part of this lost time was due to the slow motions of the participants, but also because some buttons were placed on the corners of the screen.

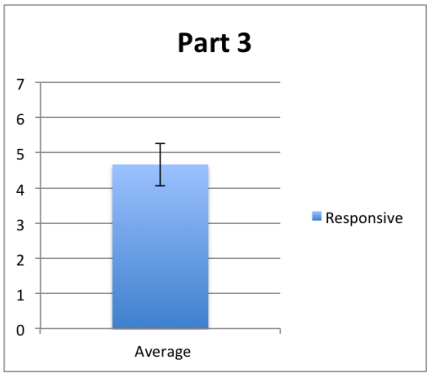
After finding out this problems, we started to realize that the user feedback was of utmost importance for our system. Also, that we were designing our GUI in a traditional way when our interaction technique is not. Thus, we decided to define the following design guidelines for our next version of the prototype.

1. **Minimize un-clickable areas**. Try to use the whole screen to display objects.

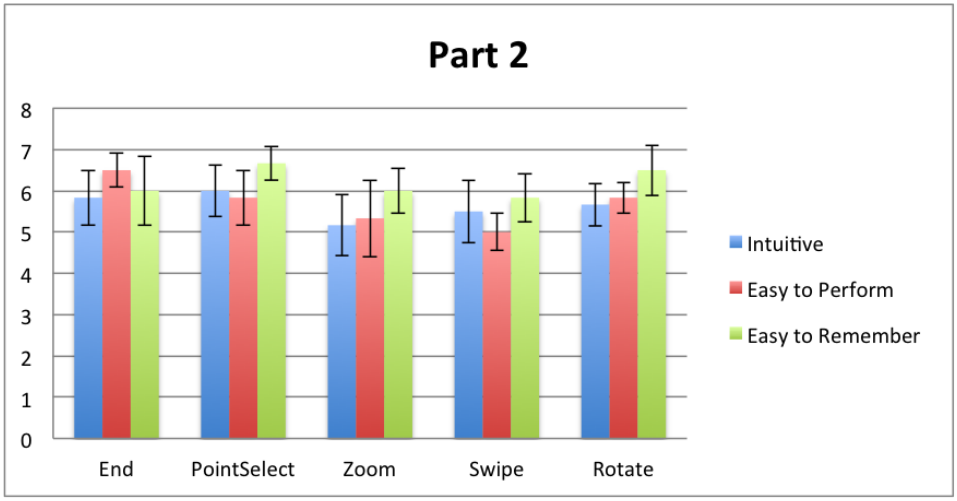
2. **Maximize accuracy**. Display the objects as big, and close to each other, as possible.



(a) Results for first part.



(b) Results for third part.



(c) Results for second part.

Figure 39: Questionnaire results for first evaluation of the system.

3. **Available gestures.** Display the available gestures to the user in the screen.
4. **Feedback in hand.** Include as much feedback as possible in the hand. This is what the user is focusing on most of the time.
5. **Everything visual.** The feedback should be visual.
6. **Clean display.** Nothing must cover the hand, and the hand should interfere as little as possible with the objects.
7. **Timing.** Make it clear to the user when the system is expecting gestures and when it is not.

Changes in the iteration. The changes to the GUI are based on the previously described problems and following the defined guidelines. Now, we describe the changes in the prototype.

a) **Screens layout.** We changed the layout of the screens making every part of them clickable.

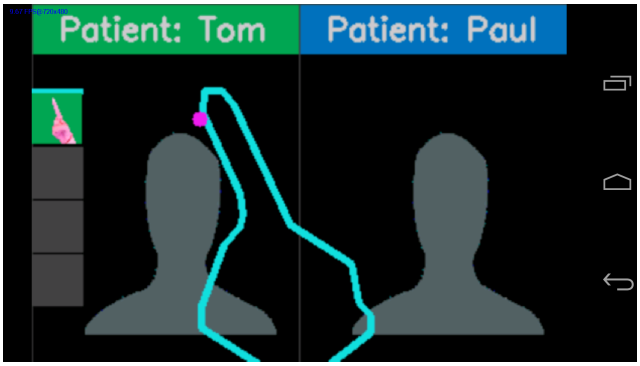


Figure 40: Here we can see some of the changes in the iteration.

b) **Contour color.** We display the color of the hand contour depending on current gesture. The default color is green.

c) **Available gestures.** We display pictures of available gestures.

d) **Current gesture.** When a gesture is detected, its picture changes the background color.

e) **Waiting period.** The hand contour is filled with red when the system is not expecting gestures.

f) **Color of aids.** Changed color of all the aids for the gestures.

g) **Zoom level.** We included a visual representation for the supported zoom levels and the current level.

h) **Number of x-rays.** We included a visual representation for the number of screens with x-rays and the current screen.

i) **Remove text feedback.** The only text feedback given is when the hand contour is too far or too close.

3.5 Summary

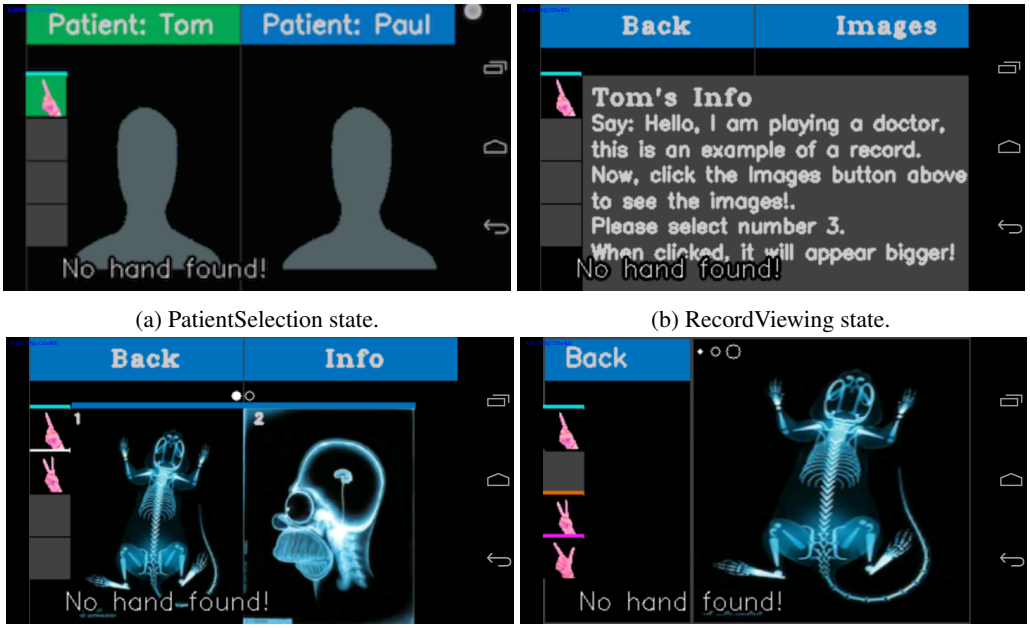
In this chapter we described the iterations of the prototype. During these iterations we took decisions regarding the gestures definition, the gesture detection method, system architecture, and GUI. In the rest of this section we first summarize the iterations. Then, we extend and present our final design guidelines. Finally, we present the final definition of the gestures.

3.5.1 Iterations

In this section we summarize the iterations of the prototype.

First iteration. During this iteration an initial definition of the gestures emerged as the result of the brainstorming session. We built rapid prototypes that led to the selection of the gesture detection method. Finally, the final software and hardware choices were taken.

Second iteration. During this iteration we first confirmed that the selected gesture detection method could be implemented in wearables and mobile devices with a mock-up application, or



(a) PatientSelection state.

(b) RecordViewing state.

(c) RecordViewing showing the patient's x-rays.

(d) ImageInteraction state.

Figure 41: Final version of the prototype.

"HelloWorld!" application. The iteration continued with our scenario definition, which led to an extension of the gesture vocabulary, the definition of requirements, and the architecture for the final prototype.

Third iteration. During this iteration we implemented the first and second versions of the prototype. The first version of the prototype was the incorporation of the ward round scenario with the gesture detection method. Internal testings and the first formative evaluation caused changes and improvements which resulted in the second version of the prototype. These changes and improvements included a change in the definition of *Init*, an addition in the waiting time after the detection of any pre-stroke (e.g. 1 second) and post-stroke (e.g. 2 seconds), the blocking of the live feed of the camera, the display of the background in black, the hand contour is shown in green, the addition of feedback to the pre-stroke of the gestures, and hover feedback to *PointSelect*.

Fourth iteration. During this iteration the results of the second formative evaluation of the prototype were analyzed. This evaluation helped us understand the feedback to the user is of utmost importance for the prototype. Then, we defined design guidelines for the GUI and increased the user feedback. Finally, the iteration ends with the final evaluation of the system.

1. First iteration
    (a) Idea
    (b) Brainstorming session
        i. First definition of gestures
    (c) Rapid prototypes were built
        i. Selection of gesture detection method

ii. Software and hardware choices

2. Second iteration
   (a) Prototype v0.0; ”HelloWorld!”
   (b) Scenario definition
      i. Second definition of gestures
      ii. Requirements definition
      iii. Architecture design
3. Third iteration
   (a) Prototype v1.0
      i. Third definition of gestures
   (b) First formative evaluation on prototype v1.0
   (c) Analysis of results from first formative evaluation: block camera feedback!
      i. Prototype v2.0
   (d) Second formative evaluation on prototype v2.0
4. Fourth iteration
   (a) Analysis of results from second formative evaluation: feedback!
      i. Design guidelines
      ii. Prototype v3.0
   (b) Final evaluation on prototype v3.0

3.5.2 Design guidelines

The design guidelines previously presented only regarded the GUI. Here, we extend these guidelines to also include the metrics followed for the definition of the gestures and our conclusions from the final evaluation of the system. With this extension we present a set of design guidelines that can be followed in the design of eyewear computer systems with hand-based gestures as input modality.

1. **Gestures**. The gestures should have the following characteristics:
   (a) *Intuitive and ergonomic.* The gestures should be easy to learn and use.
   (b) *Lack of ambiguity and ease of recognition.* The gestures should provide automatic interpretation for their use.
2. **Minimize un-clickable areas and maximize accuracy**. Try to use the whole screen to display objects and display the objects as big, and close to each other, as possible.
3. **Available gestures.** Display the available gestures to the user in the screen.
4. **Feedback in hand.** Include as much feedback as possible in the hand. This is what the user is focusing on most of the time.
5. **Everything visual.** The feedback should be visual.

6. **Clean display.** Nothing must cover the hand, and the hand should interfere as little as possible with the objects.

7. **Do not overdo it.** Do not display more information than the necessary. Avoid an overload of information.

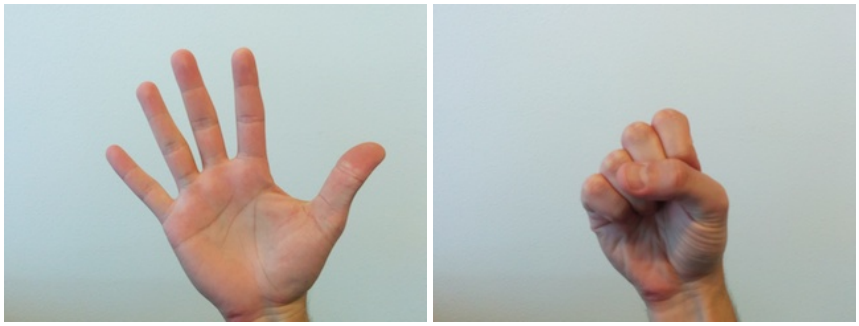
8. **Timing.** Make it clear to the user when the system is expecting gestures and when it is not.

9. **There is no rush.** Give users time to modify the hand posture by including waiting time between the detection of gestures.

10. **Leave room for error.** Give users the possibility to make errors while gesturing. In other words, don't rely only in one frame to make a classification.

3.5.3 Final definition of gestures

As describe throughout this chapter the definition of the gestures suffered at least one change in the first three iterations. Here, we summarize and present the final definition.



(a) Init gesture. (b) End gesture.

Figure 42: Overview of the static gestures supported by the system. Their main function is to control the detection of the gestures.

a) **Init.** All fingers separated and fully extended. This gesture is static (figure 42a) and it is used to start the interaction with the system.

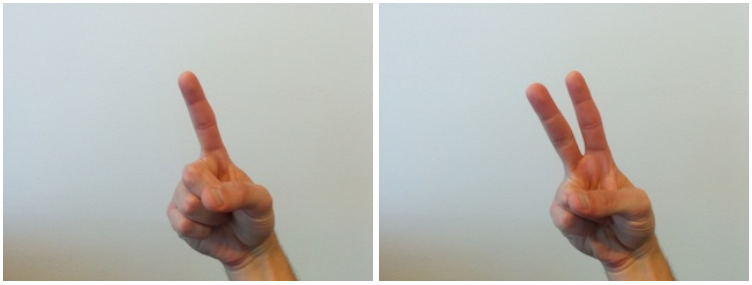
b) **End.** All fingers together and fully flexed. This gesture is static (figure 42b) and it stops the current interaction with the system.

c) **PointSelect.** Index finger lifted with the rest of the fingers tucked in. This gesture is dynamic. It selects objects displayed on screen. Its pre-stroke can be seen in figure 43a, the stroke is the finger moving around the screen changing the pointer location, and the post-stroke is lifting a second finger.

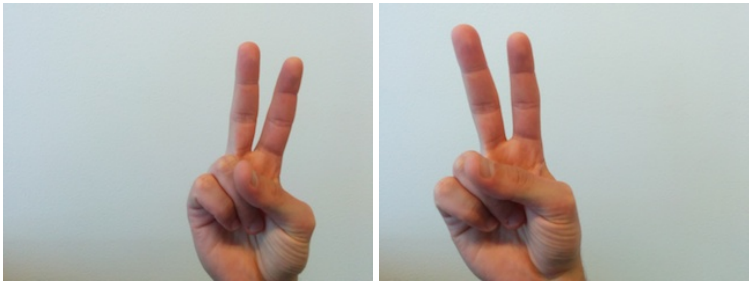
d) **Swipe.** Index and middle fingers lifted with the rest of the fingers tucked in. This gesture is dynamic. It swipes through the images displayed on screen in a carousel fashion. The pre-stroke is seen in figure 43c, the stroke is moving the hand to the left or right, and the post-stroke is the hand towards the left or right of the initial position.

e) **Rotate.** Thumb and index finger in an L-shape with the rest of the fingers tucked in. This gesture is dynamic. It rotates the image displayed on screen. The pre-stroke is seen in figure 43e, the stroke is the hand being tilted left or right, and the post-stroke is the hand titled towards the desired rotation direction.

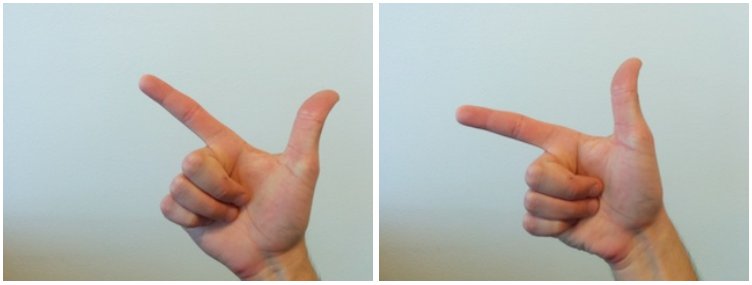
f) **Zoom.** Index and middle fingers lifted with the rest of the fingers tucked in. This gesture is dynamic. It zooms in and out the image displayed on screen. The pre-stroke is seen in figure 43g, the stroke is the fingers moving closer together or farther apart, and the post-stroke is the fingers with a smaller or bigger distance than the initial distance.



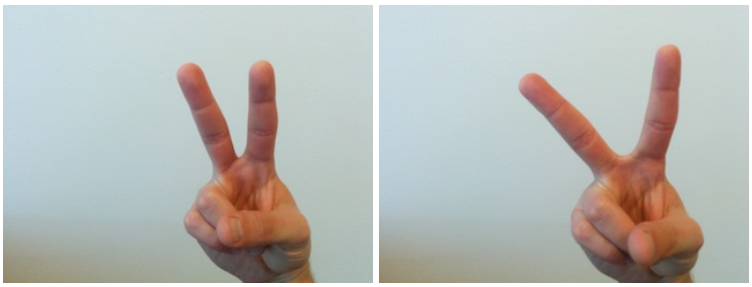
(a) PointSelect pre-stroke. (b) PointSelect post-stroke.



(c) Swipe pre-stroke. (d) Swipe (to the left) post-stroke.



(e) Rotate pre-stroke. (f) Rotate (to the left) post-stroke.



(g) Zoom pre-stroke. (h) Zoom (in) post-stroke.

Figure 43: Overview of the dynamic gestures supported by the system. The actions shown are: swipe to the left, rotate to the left, and zoom in.

4 Implementation

The previous chapter described in detail the decisions throughout the iterations regarding the design of the prototype. Here, we describe the implementation of the gesture detection method. The rest of the chapter is as follows. Firstly, we describe our gesture detection method. Our architecture is closely coupled with the Image Processing General Framework, which provides the basis for image processing in Computer Vision. With the previous, we are able to detect the gestures on the image acquired from the camera. Secondly, we describe the models for the final definition of gestures.

4.1 Gesture detection method

As previously mentioned, to be able to detect the gestures we use Computer Vision techniques. In any project using Computer Vision the Image processing General Framework can be applied (Moeslund, 2012). We now discuss how we apply this framework in our system.

Figure 44: Image Processing Framework blocks.

4.1.1 Image acquisition

For our prototype this is a very important block because here we manage to extend our robustness against light conditions. Three camera parameters [18] are set up before obtaining the frames.

Focus areas. The default focus mode for Android cameras is AutoFocus. This means that the camera driver will automatically adjust the focus. For the hand to always be in focus we change this parameter to follow its center.

Metering areas. The metering areas are used by the camera drivers to calculate the exposure. Instead of changing the settings of exposure ourselves, we set the metering area to always follow the center of the hand so the calculation of the exposure, if any, is set according to the center of the hand.

Auto white balancing. Some cameras have the option of automatically adjust the white balance in the picture and this causes the intensity of the objects in screen to change, sometimes dramatically. This parameter, if present, is set to *false* so the white in the picture stays the same and the colors in the picture are not changed.

4.1.2 Pre-processing

In this block the image is prepared for the actual processing. When we acquire the picture from the camera we use the OpenCV operation *pyrDown* (Solem, 2012), which gives a picture half the size of the original. We then convert the color space to HSV, because it is an optimal space for segmentation of objects (Sural et al., 2002). The properties, e.g. saturation and value, of this color space allow for the segmentation of the color to adapt better against changes in light conditions.

[18] http://developer.android.com/reference/android/hardware/Camera.Parameters.html

4.1.3 Segmentation

Following the pre-processing comes the Segmentation. For this block we use the simple operation *inRange* from OpenCV (Solem, 2012). This operation returns the pixels that fall within the given range, leaving the rest in black. To make the detection of the hand easier, we complement this technique by using gloves of a distinctive color as seen on figure 45.

Figure 45: Here we can see the glove being used to improve the performance of the prototype.

Having the corresponding blobs, we apply morphology operations to them. Specifically, we apply the *open* operation (Solem, 2012). This operation first gets ride of small noisy blobs in the image and then fills any holes in the bigger blobs. Once we get rid of the noise we extract the contours in the image and iterate through them. A contour is a list of points that define a curve in an image (Bradski and Kaehler, 2008). In our case this curve encloses the hand. We pick the biggest one as the hand. We choose to do this distinction since the hand of the user is supposed to be the only object of that color in front of the screen, also the object that covers the screen in the largest scale. After selecting the biggest contour we filter it again by size but in a different perspective. For the best performance of the gesture detection the hand needs to be at a certain distance, or have a certain size in the image. This is why we add the constraint for the contour to be bigger than the 5% of the screen area and smaller than the 28% of the screen area. Having the hand contour selected we calculate its center and convexity defects. Convexity defects (figure 46) are a way of characterizing the shape of an object (Bradski and Kaehler, 2008). A visual representation of the defects in the gestures can be seen in figure 47.

4.1.4 BLOB analysis

The BLOB Analysis block classifies the available contours. In our case, the contours are classified by their convexity defects. After our gestures were defined, their characteristics and hand pose was studied and the result is a model of each gesture based on the defects of the hand, e.g. fingers, and their characteristics. We define two types of types of features for the models: general features, and specific features. The former regards the number of lifted fingers, and the latter regards specific characteristics of each gesture. Our classification has three steps: filtering, comparison of general features, and comparison of specific features.

Filtering. After calculating the convexity defects in the segmentation block we start by filtering them. This filtering process allows us to be certain that the remaining defects represent the fingers of the hand. The filtering of the defects is done according to the following criteria.

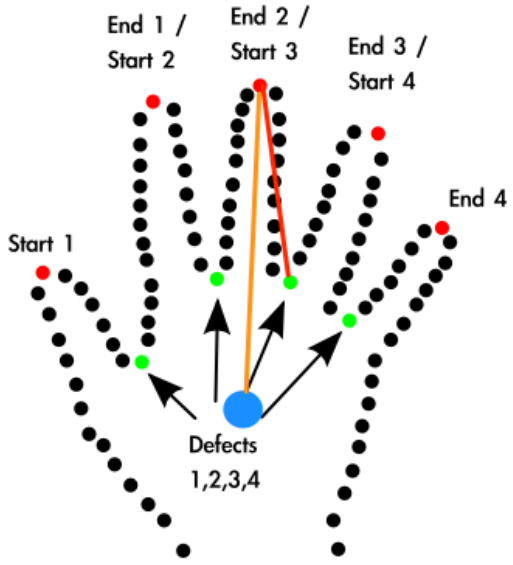


Figure 46: Convexity defects. The red dots represent the start/end of defects, the blue dot represents the center of the hand, and the green dot represents the defects itself.

1. The *start point* of the defect should be above the center of the contour.
2. The *farthest point* (green dot) of the defect should be above the center of the contour.
3. The *farthest point* (green dot) of the defect should be below the *start point* of the contour.

At this point we are sure that the remaining defects represent the fingers of the hand in screen. Before the analysis and classification starts, we map the upper part of the screen to the entire screen with a transfer function. Here, we take the first defects and calculate the shifting distance $s = y * scaleFactor - y$; where y is the y-coordinate of the defect and $scaleFactor = 2.5$. The scale factor was acquired empirically in internal tests. Once the shifting distance s is calculated, we iterate through all the defects and sum $y_i = y_i + s$. Once the convexity defects are filtered and shifted, we can analyze and classify them.

Comparison of general features. In this step we compare the defects against the general features of the models of the gestures. This comparison already gives a possible classification of the defects. We take the number of lifted fingers and decide the possible gesture being performed. For example, if the number of defects is one the only possible gesture being performed would be *PointSelect*.

Comparison of specific features. After a possible gesture is selected, we follow by comparing the defects agains the specific features of the gestures. As seen on figure 46, each defect is composed by a starting point, an ending point, and the defect itself. In the same figure we also depict the center of the contour. We take 1) the start of the defect 2) the defect and 3) the center of the hand, and calculate the following five values that provide the basis for the specific features of the gestures.

1. *distanceStartCenter*: distance between the start of the defect and the center of the hand. Orange line in figure 46. This value is used for all gestures.

2. *distanceStartFarthest*: distance between the defect itself and the start of the defect. Red line in figure 46. This value is used for all gestures.

3. *relationOfDistances*: relation between *distanceStartCenter* and *distanceStartFarthest*. It is calculated by diving *distanceStartCenter* over *distanceStartFarthest*. We chose to use this relation in order to avoid using fixed values. In this way, the calculated values behave like a model of the characteristics and can be applied to a wider set of hands. This value is used for all gestures.

4. *distanceBetweenDefects*: distance between the start of two defects. This value is used for *Zoom* and *Rotate*.

For the following description we recommend the reader to focus on figure 46. We start by making sure that the *relationOfDistances* is less than two. That is, *distanceStartCenter* should be less than two times the *distanceStartFarthest*. This value is greater than 2 when a finger is folded or when the fingers are closed together. This applies for all the gestures since we defined them with fully extended fingers separated from each other.

At this point, the static gestures *Init* and *End*, and the dynamic gestures *Swipe* and *PointSelect* can already be classified. On the other hand, *Zoom* and *Rotate* need further analysis because they are part of the same overall state, e.g. **ImageInteraction**. We use two values to tell them apart. For *Zoom*, the value *distanceBetweenDefects* should be **less** than *distanceStartCenter*. For *Rotate*, the value *distanceBetweenDefects* should be **more** than *distanceStartCenter*.

4.2 Gestures models

As previously mentioned, after the definition of the gestures we analyzed them in order to make a model of the characteristics of their defects. Here, we present the model for each gesture.

a) *Init.* Three or more defects representing the fingers fully extended.

b) *End.* Three or more defects representing the fingers fully flexed.

c) *PointSelect*

   (a) **Pre-stroke.** One defect representing the index finger.

   (b) **Post-stroke** Two defects representing the index and middle fingers.

d) *Swipe*

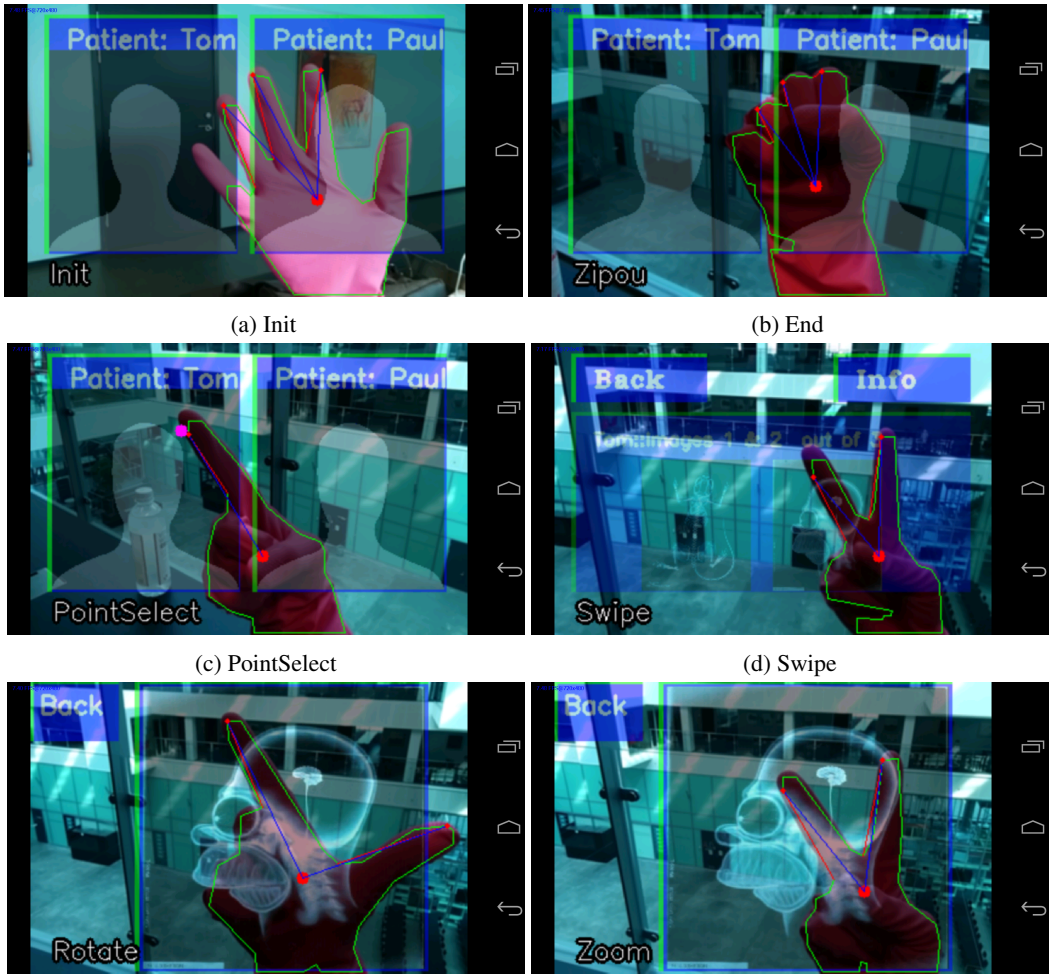
   (a) **Pre-stroke.** Two defects representing the index and middle fingers. The middle point between the defects is taken as the initial position of the gesture.

   (b) **Stroke.** Same pose as the pre-stroke with the hand moving across the screen.

   (c) **Post-stroke.** Same pose as the pre-stroke with the hand moved across the screen. The middle point between the defects is taken as the final position of the gesture. The distance between the initial and final positions is, at least, 20% of the screen's width.

e) *Rotate*

   (a) **Pre-stroke.** Two defects representing the index finger and thumb. The distance between the defects has to be greater than the their distances to the center of the hand. The middle point between the defects is taken as the initial position of the gesture.



(a) Init (b) End

(c) PointSelect (d) Swipe

(e) Rotate (f) Zoom

Figure 47: Here we can see the first version of the prototype before the first formative evaluation and the model of the gestures with their special characteristics. The pre-strokes are shown in gray, and the post-strokes are shown in purple.

(b) **Stroke.** Same pose as the pre-stroke with the hand being tilted towards the direction of rotation.

(c) **Post-stroke.** Same pose as the pre-stroke with the hand tilted towards the direction of rotation. The middle point between the defects is taken as the final position of the gesture. The distance between the initial and final positions is, at least, 10% of the screen's width.

f) *Zoom*

(a) **Pre-stroke.** Two defects representing the index and middle fingers. The distance between the defects has to be less than their distances to the center of the hand, and is also

set as the initial distance between defects.

(b) **Stroke.** Same pose as the pre-stroke with the fingers coming closer together or farther apart.

(c) **Post-stroke.** Same pose as the pre-stroke with the fingers closer together or farther apart. The distance between the defects is taken as the final distance between defects. The difference between the initial and final distances between the defects has to be less than, or greater than, 5% of the screen's width.

5 Evaluation

From our goals, defined in section 1.1.1 (page 5), we have developed a hand based gesture recognition system for an eyewear computer using its camera, we have defined a set of gestures, and implemented a gesture detection method optimized to detect the set of gestures.

The second formative evaluation showed us that the implementation of the prototype for an eyewear computer has its challenges. Although it works perfectly, the hardware shows struggling signs with extended periods of use of our prototype. These challenges are addressed in section 6.2.1. Here, our final goal of evaluating the intuitiveness of the gestures, the user experience, and the precision and recall of the gesture recognition is accomplished.

In this chapter we address the final evaluation of the system. In this evaluation we focused on three aspects. First, a formal evaluation of *PointSelect* as a pointer mechanism. It was not envisioned at the beginning of the project to have a pointer mechanism. The definition of the ward round scenario included *PointSelect* to the set of gestures and it started acquiring more importance over time. That is why we include the evaluation of *PointSelect* as a pointer mechanism. Second, the performance of the gesture detection method. We evaluate the performance of the method by measuring its precision and recall with a confusion matrix. Third, the intuitiveness of the gestures and the user experience. These aspects will be evaluated with a questionnaire. Although so far we have stated that we evaluate the intuitiveness of the gestures, we extend the evaluation according to our design guidelines and evaluate the intuitiveness, ergonomics, ease of recognition and lack of ambiguity of the gestures.

From the second formative evaluation we learned that the participants needed more time to learn the gestures and how to perform them. Thus, we defined a formal training phase to get the participants closer to an expert level. Also because of our experience during the second formative evaluation, this final evaluation was designed to be held in a Nexus 5 smartphone.

5.1 Design

We recruited 9 participants (age average=29, $\sigma = 6.4$, 1 female) for the evaluation. 2 participants had intermediate compute skills, and the rest were expert computer users. One participant had myopia, and one participant used glasses. 2 participants had previous experience with smart glasses, and all the participants were right-handed. All the sessions were recorded in video. The evaluation was designed to take place in three phases. We omit Init from the evaluation since the scenario was not designed to include a 40 seconds waiting period.

5.1.1 First phase

This was the training phase, we designed this part more seriously compared to the second formative evaluation. Here, the participants used an application very similar to the prototype for more than 10 minutes. The goal of this phase was to get the participants closer to an expert level.

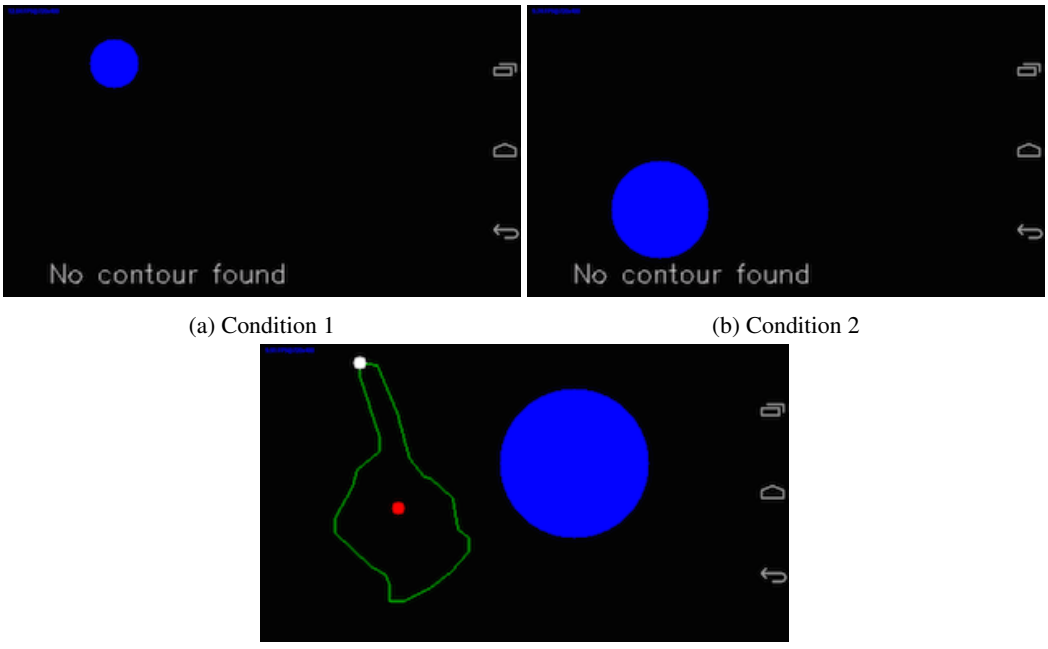
5.1.2 Second phase

For the second phase the participants were asked to follow a similar scenario to the ward round scenario presented in chapter 1 (see page 6). The participants went through the scenario one time as a practice round. The actions in the scenario to perform by the participants are described next.

1. Select patient Tom.

2. Read the first sentence on screen. After that sentence there are some more instructions. Please follow the instructions. (Select image 3)
3. Rotate the selected image upside-down.
4. Zoom in the image twice.
5. Navigate through the image to find the red dot. The dot does not have to be centered, only inside the display.
6. Zoom out the image twice.
7. Go back to the patient record.

After the scenario, the participants were asked to answer a questionnaire. The questionnaire was the same as in the second formative evaluation and can be seen in Appendix App:firstQuestionnaire. The questionnaire had three parts. In the first part we asked the participants to identify the gestures in a table by matching the corresponding name with the gesture's picture. In the second part we asked three questions 1) How intuitive are the gestures? 2) How easy are the gestures to perform 3) How easy are the gestures to remember?. The questions have a 7-point Likert scale ranging from the most negative, e.g. not intuitive, to the most positive, e.g. very intuitive. And in the third part we ask the user about the responsiveness of the system. The question has a similar 7-point Liker scale range as the previous questions.



(a) Condition 1

(b) Condition 2

(c) Condition 3

Figure 48: Conditions for the second *PointSelect* experiment.

5.1.3 Third phase

The third phase of the evaluation involved an experiment with *PointSelect*. This time, the experiment was designed more seriously compared to the second formative evaluation. Fitts' Law refers to the time a person needs to point at a target of a given size in a given distance (Drewes, 2013). It states that it takes more time to hit a target the farther and/or the smaller it is. Thus, if Fitts' Law is proven, it means that the time it takes to select an object on screen can be predicted.

The goal of this experiment was to prove Fitts' Law with *PointSelect* as the pointing device. For this we designed an application with 3 conditions. Each condition had 10 circles of 20, 40, and 60 pixels , respectively, appearing in a random location in the screen one by one. The task for the participants was to **only** point at them, not click them. The participants were shown from 2 to 3 circles for them to fully understand the task.

5.2 Results

Overall we believe that the training phase had a direct impact and allowed the participants to have a better performance in the completion of the scenario. It also gave the participants more exposure to the gestures, which we believe gives better grounds for the questionnaire's answers.

5.2.1 Questionnaire.

The results of the questionnaire (figure 49) improved compared to the first evaluation. The only mediocre value was the easiness to perform *Zoom*.

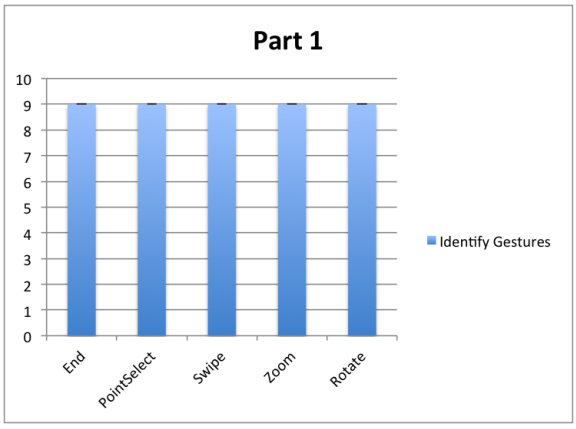
Part 1 had perfect results with the 9 participants correctly identifying every gesture. This confirms that the gestures definition follows our design guideline of *lack of ambiguity and ease of recognition*.

The results in part 2 confirm that the gestures are *intuitive*, easy to remember, and most of them *ergonomic*. This time the participants rated *Zoom* as not *ergonomic*. Although the definition of the gesture did not change, we believe this was caused by the extended time of use of the gestures. Compared to the previous evaluation were the participants gestured for about 5 minutes, this time the participants actively gestured from 10 to 15 minutes which may have led to a fixation towards *Zoom* and its difficulty to be performed.

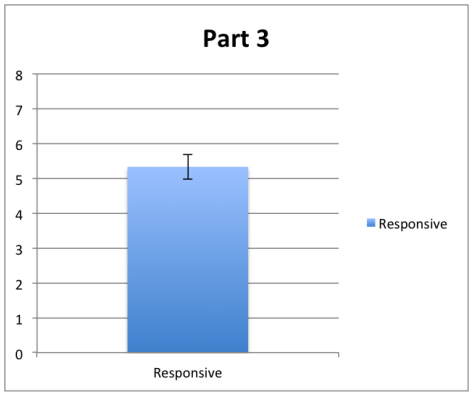
The results in part 3 show that the responsiveness of the system increased a little compared to the second formative evaluation ($\bar{x} = 4.6$) with an average of 5.3 ($\sigma = 0.71$). We thought the increase of user feedback would make the responsiveness to rank higher, but it did not.

Gesture	Intuitive	Easy to Perform	Easy to Remember
End	$\bar{x} = 6.7, \sigma = 0.4$	$\bar{x} = 7.0, \sigma = 0.0$	$\bar{x} = 6.8, \sigma = 0.3$
PointSelect	$\bar{x} = 6.7, \sigma = 0.4$	$\bar{x} = 6.5, \sigma = 0.7$	$\bar{x} = 6.8, \sigma = 0.4$
Swipe	$\bar{x} = 5.1, \sigma = 1.8$	$\bar{x} = 5.4, \sigma = 1.6$	$\bar{x} = 6.1, \sigma = 1.1$
Zoom	$\bar{x} = 5.0, \sigma = 1.1$	$\bar{x} = 3.6, \sigma = 1.9$	$\bar{x} = 6.0, \sigma = 1.0$
Rotate	$\bar{x} = 5.7, \sigma = 0.8$	$\bar{x} = 6.0, \sigma = 0.8$	$\bar{x} = 6.3, \sigma = 1.2$

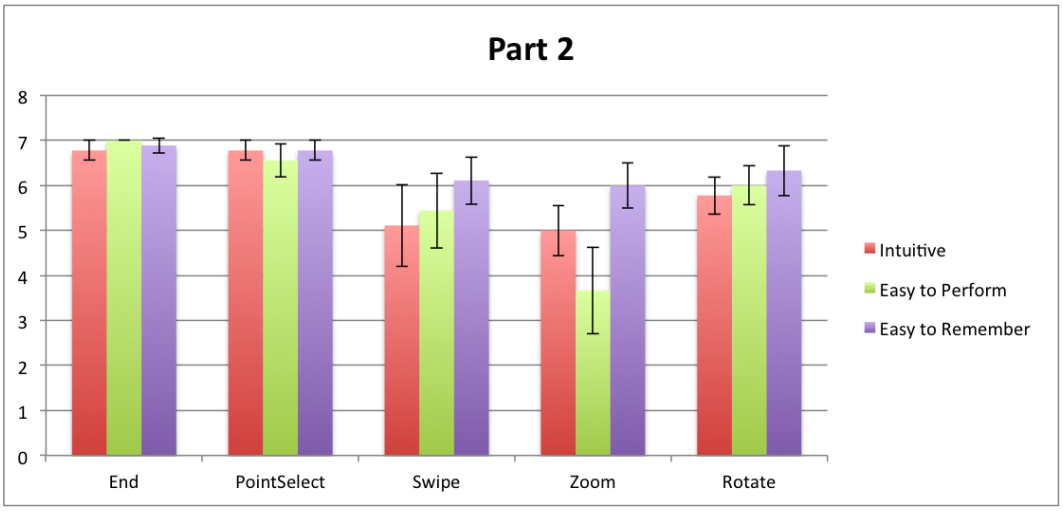
Table 4: Results for second part of the questionnaire.



(a) Results for first part. Every participant identified correctly each gesture.



(b) Results for third part. Average responsiveness of 5.3 ($\sigma = 0.71$)



(c) Results for second part.

Figure 49: Questionnaire results for second evaluation of the system.

5.2.2 Gesture detection performance

Even though it is not within the scope of this thesis to present a novel gesture detection method, measuring the performance of our method complements the results and may uncover factors not present in the questionnaire.

With the recorded videos of the second phase of the evaluation, we summarize the classification of the gestures in a confusion matrix. We consider a classification correct from the system's point of view and not the user's. We measure the performance of our gesture detection method calculating the precision and recall of the classification. Precision (equation 1) refers to the fraction of correct classifications, and recall (equation 2) refers to the fraction of classifications relative to the number of gestures performed (Han, 2005).

	Classified gesture					
Actual Gesture	PointSelect	Swipe	Zoom	Rotate	End	idle
PointSelect	86					19
Swipe		9				
Zoom	1		59			1
Rotate	3			19		2
End	2				15	
Idle						

Figure 50: Confusion Matrix for the gesture detection in the scenario. The values below the white boxes are false positives, the values to the right are false negatives. The values in white are true positives.

$$P = \frac{\text{true positives}}{\text{true positive} + \text{false positives}} \tag{1}$$

$$R = \frac{\text{true positives}}{\text{true positive} + \text{false negatives}} \tag{2}$$

The performance of the gesture detection method also improved with a 96.91% precision and a 89.52% recall. Compared to the second formative evaluation, now we can see that the number of false positives (figure 50) decreased even though the number of participants increased. The improvement of the detection method performance is a result of bug fixes in the prototype, but we also believe that the training session played a part. Now that the participants had more time to perform the gestures and use the training application, they had a better idea on how to perform them and how the system expects the gestures to be performed. These results are further discussed in the next chapter (page 65).

5.2.3 *PointSelect* experiment

A typical Fitts' Law evaluation plots execution time against IDs complemented with a regression test to show if there is a linear dependency (Drewes, 2013). The IDs are the index of difficulty for each circle defined as $ID = log_2(A/R+1)$ where A is the distance between the initial position and the center of the target, and R is the radius of the target. For the regression test the plotted data is fitted to the equation $T = a+b*ID$ producing a trend line with a regression factor or correlation R^2. Where T is the time necessary to hit the target, a is reaction time, and b is the time to process one bit by the human nervous system. For Fitts' Law to be proven R^2 needs to be close to 1.

In our experiment, we gathered 270 samples (3 conditions, 10 circles per condition, 9 participants). Our regression equation is $y = 638.19x - 1.71$ and $R^2 = 0.236$. The results for this experiment (see figure 51) show that our pointing gesture does not follow Fitts' law. In other words, we can not predict the time it takes the user to point at a target in the screen based on its distance and size, which further motivates our design guidelines **Minimize un-clickable areas** and **Maximize accuracy**. A discussion on the results of this experiment follows in the next chapter (page 65).

After reviewing the videos from the experiments we notice that the participants always take the same time to point at objects in the screen. Their movements are careful and slow no matter the size of the object. This is further discussed in the next chapter (page 69).

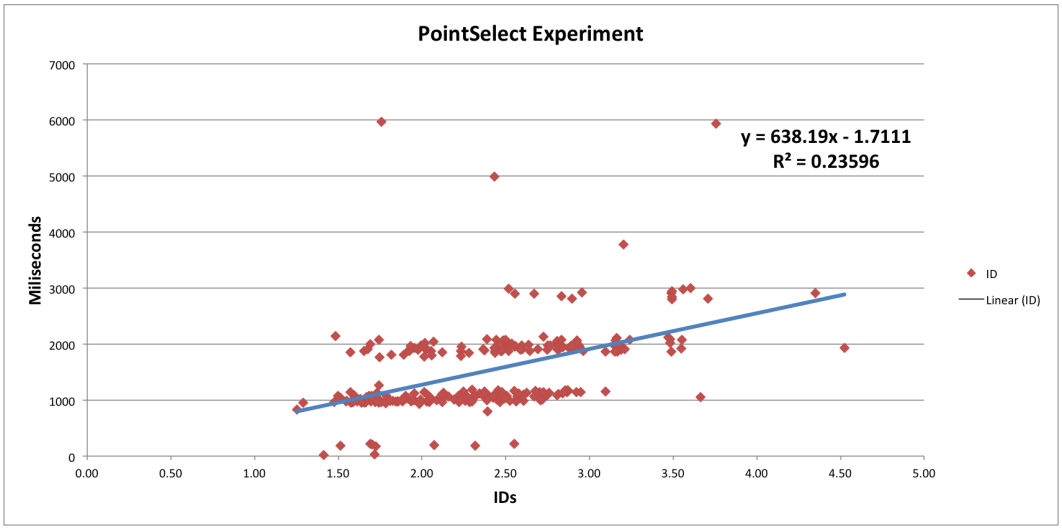


Figure 51: Results of the *PointSelect* experiment showing 270 samples (3 conditions, 10 circles per condition, 9 participants). The graph shows IDs plotted against milliseconds. The regression equation is $y = 638.19x - 1.71$ with $R^2 = 0.236$.

6 Discussion

Our final evaluation shows that 1) the gestures are intuitive, easy to recognize, and most of them ergonomic, 2) a 96.91% precision and a 89.52% recall of the gesture detection method, and 3) we can not predict the time to select an object based on its size and distance from the pointer. In this section we discuss these results, the drawbacks of the prototype, and changes that would enhance the interaction. The rest of the section is as follows. Firstly, we discuss the results of the final evaluation and the results of similar systems. Secondly, we discuss the challenges of the hardware. Next, we discuss the gesture detection method and suggest future improvements. Finally, we discuss and suggest improvements for the user feedback.

6.1 Final evaluation results

The results of the final evaluation gives us two important points to discuss. The first point is the design of the *PointSelect* experiment, and the second is the result of the gesture detection method performance. Firstly, we address the results of similar systems, then we follow with the discussion of the two points.

6.1.1 Results of similar systems

As previously mentioned, only four projects presented in section 2 refer to aspects like usability and user experience.

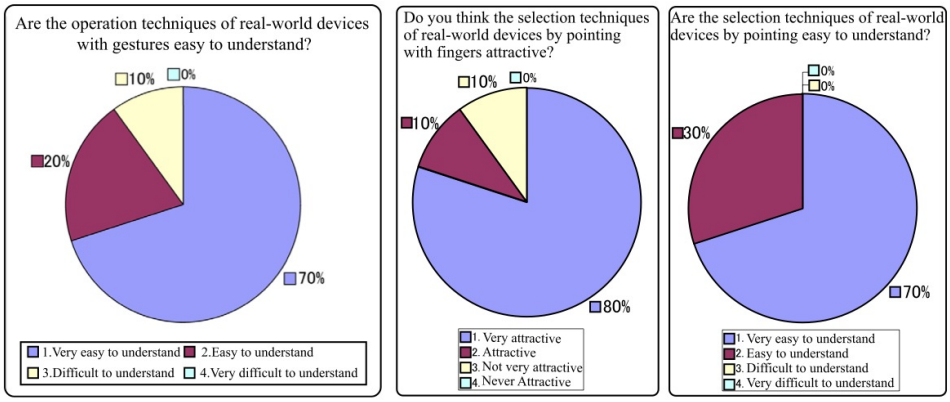


Figure 52: Ubi-Finger questionnaire results.

Tsukada and Yasumura present the results of a questionnaire with three questions applied to the participants in their evaluation. The questions are: Are the gestures easy to understand?, Is the selection o real-world devices by pointing with fingers attractive?, and Is the selection of real-world devices by pointing easy to understand?. Each question has four answers from the most positive, e.g. very easy to understand, to the most negative, e.g. very difficult to understand. They report that 1) 70% of the participants thought their gestures are very easy to understand, 2) 80% of the participants thought the selection of real-world devices by pointing with fingers is attractive, and 3) 70% of the participants thought the selection of real-world devices by pointing is easy to understand (Tsukada and Yasumura, 2002). We can not directly compare our results with them since we use a 7-point Likert scale and they use some sort of 4-point scale. But considering that the easiness to

understand a gesture is the same as an intuitive gesture, we can say that we achieve similar results in both sets of gestures.

Kim et al. present a different kind of results. They reach three conclusions. The first conclusion is, even though participants found that using a gesture is a good way to control devices, users are required to carefully define their gestures in order to avoid false-positive errors during use. The second conclusion is that their system can make activities more engaging. The third conclusion is that it could be a good platform for social interaction (Kim et al., 2014). We do not believe we can relate to any of the results.

Argyros and Lourakis compare their sets of gestures (2D vs 3D). They reach the conclusions that 3D gestures are more easy to understand and assimilate, the users found the 3D implementation of mouse clicks (hand motion towards the camera then towards the user) more intuitive because it is analogous to pressing an imaginary 3D button, and that the 2D gestures are more responsive and therefore much more user-friendly (Argyros and Lourakis, 2006). Since we do not support 3D gestures we can not relate directly to this results.

The findings of Jalaliniya et al. include that the combination of foot and hand gestures is an acceptable alternative to interact with images without direct contact. Only two of their questions relate directly to our questionnaire. The participants in this study found the hand gestures intuitive for interaction, and they also found the system responsive (Jalaliniya et al., 2013). If we take an overview of the intuitiveness results for our gestures, we can say that we reach a similar result. On the other hand, they show a better results for responsiveness. The results of our responsiveness are 5.3 in a 7-point Likert scale. With a 5 five-point Likert scale where 5 is strongly agree, their result is above 4.

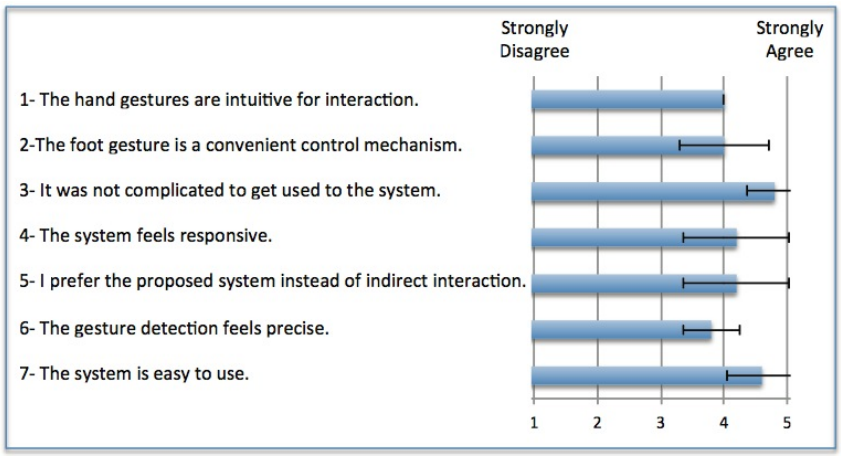


Figure 53: Touch-less Interaction with Medical Images using Hand & Foot Gestures questionnaire results.

6.1.2 *PointSelect* experiment

A typical Fitts' Law experiment includes pointing **and** clicking of the pointing device. The typical number of target sizes is 4 with the ratios of 1, 2, 4 and 8. Also, the different distances between targets are fours with the same ratios. At the time we designed the experiment we overlooked this

factors and designed the experiment in a different manner. Firstly, instead of point and click we only used pointing because we though that including the post-stroke of *PointSelect* would introduce noise to the experiment. Secondly, instead of 4 different target sizes with ratios of 1, 2, 4 and 8 we used 3 target sizes with ratios of 1,2 and 3. Finally, we assigned random distances between the targets without any clear ratios. In the results of the evaluation (page 63) we concluded that our pointing gesture does not follow Fitts' Law, but another reasonable conclusion could be that the experiment is inconclusive, even if the results were positive, since the experiment did not follow the typical design.

6.1.3 Gesture detection method results

In order to measure the performance of the detection method we took a classification as correct from the system's point of view, not the users. For example, while using *Rotate* one participant started with the hand in a completely horizontal position (e.g. the index finger in a horizontal straight line pointing towards the left of the screen). This is a good set up to rotate the image to the right, but the participant tried to rotate to the left and the system was not detecting the rotation because the defect for the index finger was below the center of the contour. The participant tried to go back to the initial position, but instead rotated the hand more to the right of the initial position causing a classification of the rotate-to-right action.

In the previous example we can make the conclusion that the movement was correctly classified by the method as a rotation to the right, even though the user was trying to rotate to the left. We believe this situation gives a false illusion in the results of the performance. In other words, even though we show a 96.91% precision and a 89.52% recall, from the user's point of view that may not be true. This is further discussed in section 6.3.1.

6.2 Hardware

In this section we address the challenges of the hardware, and how we encountered them.

6.2.1 Power and heat

Starner (2001) presents many challenges in wearable computing. Here, we address power use and heat dissipation. The former as the most limiting factor in mobile technology, and the latter as a companion of the former.

During our design process and internal testings we did not encountered these problems. But we encountered both challenges during the second formative evaluation of the prototype with the increase of the time of use. During the first session, the M100 where heating up and powering off after extended periods of use. The first time it happened was shortly after 10 minutes of use but every time we restarted the glasses without letting them rest it kept happening more frequently. This caused the rest of the sessions to have the Nexus 5 smartphone as the hardware, instead of the M100. We learned from this situation and designed the final evaluation to be held with the smartphone and the M100 could be used at the end of the session as an extra feature.

6.2.2 Computational cost

Wilson (2007) addresses the computational cost challenge stating that, although processors keep getting smaller and wearable devices are becoming more powerful, image processing algorithms still show to be computationally intensive for wearables. We faced this challenge continuously throughout our project. It may even have helped the M100 to heat faster during the evaluations.

Based on the rapid prototypes we developed during the first iteration, we selected the gesture detection method based on its computational cost. Compared to the rest, it was the lightest method. If computational cost was not an issue, we could have implemented other techniques, e.g. HMM, and support a different type of gestures. For example, we could have defined the rotate gesture as one finger ”drawing an U” in the screen, as suggested in the brainstorming session. Also, when we selected the hardware for the final implementation it was based on this aspect. The M100 showed a better performance over the Google Glass.

But this was not an issue only in the beginning. As the prototype started growing and evolving, the performance started dropping. We had to be careful with the modifications or inclusions to the design because it was very easy for the frames per second to drop, causing the objects displayed in the screen to be delayed by seconds instead of being displayed in real-time.

6.3 Gesture detection method

As previously mentioned, even though our gesture detection method shows a good performance, we think it might be improved. Here, we discuss the drawbacks and possible improvements of the method.

6.3.1 Classification

When testing internally, our detection method works well because we make the gestures in a very explicit way and following the models very precisely, after all we defined the models, but the users didn't make the gestures in the same way. We could see this happening specially with *Zoom* during the final evaluation of the system. Some participants seemed to be doing the gesture in a very awkward way for the system that resulted in wrong classifications from the user's point of view, e.g. zoom-in instead of zoom-out. The previous example can be seen in figure 50 where there is a high number of *Zoom* classifications, or true positives. After some analysis we concluded that during the definition of the gestures models we did not take into account the noise of the defects. That is, the defects are not always placed at the tip of the fingers, sometimes they are found on the sides as seen in figure 54. Thus in the previous example, the participants gestured *Zoom* in a way that, without moving the hand too much, the defects moved enough for zoom-in to be classified. Instead of changing the model for the gestures, an improvement could be processing the defects instead of just filtering them. Maybe a weighted average would be a good idea to have more stable defects. This would have had a big impact in *Zoom*, were the participants had the tendency to make small movements instead of explicit movements. this may improve the recall of the detection.

Another drawback in our detection method is that we base the classification in only one frame. In the example given for the confusion matrix results, the hand postures of the participant caused the system to classify rotation to the right although the participant was trying to rotate to the left. Analyzing this example we come to the conclusion that the classification in our gesture detection method could be improved, not because it is wrong, but because there needs to be room for errors. Although the classification is correct, the participant did not mean for that gesture to be classified. One way of improving this behavior could be to include a timer or frame counter before classifying a gesture like the one presented by Davis and Shah (1994) where they wait at least 3 frames to detect motion. In other words, user's errors need to be accepted. This may decrease the recall, but it would show a more truthful precision of the method from the user's point of view.

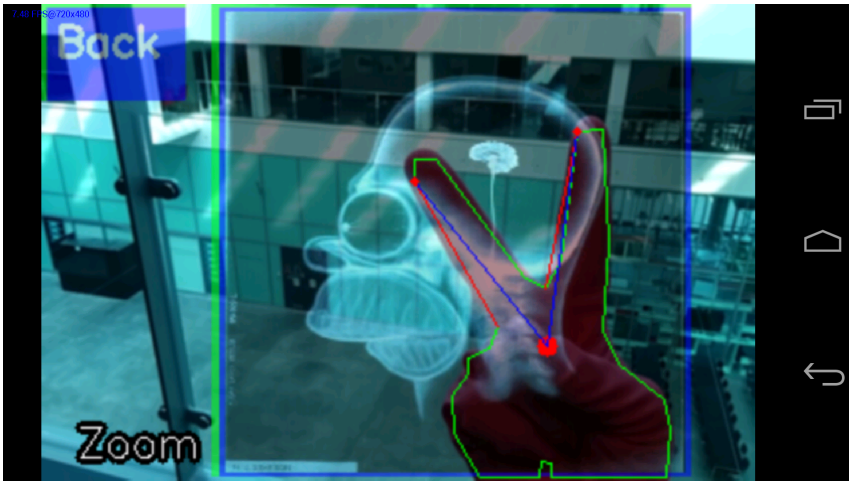


Figure 54: *Zoom*'s defects not alined with the tip of the fingers.

6.4 Gestures

In this section we first address the challenges of *PointSelect* as a pointing device. Then, we address the definition of *Zoom* and *Swipe*.

6.4.1 *PointSelect* as pointing mechanism

After reviewing the videos from the experiments we notice that the participants always take the same time to point at objects in the screen. Their movements are careful and slow no matter the size of the object. Analyzing this behavior and a comment from one of the participants during the final evaluation, we have come to the next conclusion. Our pointer is shown on a 2D screen but the movement of the hand has 3 dimensions. The user can move from left to right and towards or further away from the camera. The latter has an special importance because if the user is too far or close the hand will not be detected. Thus, adding the third dimension to the pointing movement makes it difficult for the user to control it in a 2D display. Besides the design factors already discussed, this conclusion is also a factor in the results of the *PointSelect* experiment, where we conclude that our pointing gesture does not follow Fitts' Law. Both, the results of the experiment and the conclusion, confirm even further one of our design guidelines. When designing a user interface for hand-based gestures as input modality the clickable objects need to be as close as possible and cover as much of the display as possible, e.g. **minimize un-clickable areas and maximize accuracy**.

Another drawback of *PointSelect* was the speed of the pointer. After the first formative evaluation we removed the camera feed and displayed the hand contour in green with the background in black in order to map the upper half of the screen to the entire screen. For the mapping we used a transfer function applied to each defect right before they are classified. The previous was done with every gesture, but here we focus on the effect it has on *PointSelect*. Because of its definition, the transfer function makes the defects to shift in a greater scale the lower they are. This gives the appearance that the hand move faster the lower it goes. This seemed to cause some problems for the participants in the final evaluation making it harder to control the pointer on the bottom part of the screen. Thus, a better transfer function would be a good addition to the prototype.

6.4.2 Ergonomics, ease of recognition and lack of ambiguity

When *Zoom* was defined, we tried to emulate the way people zoom in images on their smartphones, e.g. for zoom-in one would pinch the screen with the index finger and thumb together and move them further apart. We thought the movement was very intuitive and easy to make, but the latter turned out not to be true. In the final evaluation the participants gave the gesture a low score when they were asked if the gesture was easy to perform. After analyzing the videos of the sessions we came to the conclusion that it is not the pre-stroke the one that is difficult to perform, but the post-stroke is the one causing problems. Thus, a better definition of the gesture could be done for it to be easy to perform and follow our *ergonomic* principle. For example, instead of moving the fingers closer or away from each other, the stroke could be defined as moving the whole hand - not the fingers - towards or away from the screen. Although it was not intended, the previous example is actually possible with the current prototype but at the time of the evaluation we had not realized it.

When we defined *PointSelect* in the second iteration we had to change the definition of *Swipe*. We considered the scenario and came to realize that it was possible to use the same pre-stroke's characteristics as *Zoom* because they are used in different overall states and we thought it was also intuitive. If we analyze this definition thinking only on the detection method, which was the case, it is in fact possible to have both gestures with the same pre-stroke's but we did not follow our own criteria for the definition of the gestures. The decision of defining both pre-strokes with the same characteristics went against our **ease of recognition and lack of ambiguity** principles. Although the results of the final evaluation show that the participants did not find them confusing, one of the participants in the second formative evaluation did get confused and selected *Zoom* twice while identifying the gestures. Because of time constraints and the maturity of the prototype we did not modify these gestures at the time but a better definition of either one of them could be achieved.

6.5 Feedback

Previous experiences with web and mobile development thought us that feedback is an important aspect in any interactive application, but the importance of the user feedback in gesture recognition shows to be greater and took us by surprise. Here, we first address the feedback of the hand contour. Then, we address auditory feedback as an improvement for the prototype.

6.5.1 Hand contour

Mapping the upper part of the screen to the entire screen and showing the hand contour in green showed to be a good aspect of the prototype. For example, showing the contour in the screen gave the user the necessary feedback of where and how far to place the hand. When the participants in both the second and final evaluation did not see the contour, they started moving the hand towards the center and closer to the camera. The previous corresponds with an example given by Wilson (2007), where a game ensures that the players stay in the camera's field of view by displaying them in the screen, much like we display the hand of our user's in the screen.

But it also had some downsides. During the final evaluation we encountered a special situation. When one participant was trying to click the right-bottom part of the screen most of the hand was not showing, nothing was being detected, and this made it difficult for the participant to know what was going. This caused close to 15 "Idle's" in the confusion matrix (figure 50) instead of one classification of *PointSelect*. After analyzing the videos we concluded the following. In this particular case, some part of the hand was outside the camera's field of view, but the hand contour was big enough to be detected. This combination caused the hand to be detected, but its defects were not correct. Thus, the hand was detected but the gesture was not classified. An improvement for the

prototype would be to help the user in those situations. ”If possible, the system should provide some indication of the nature of any failure so that the user can modify his or her own behavior to meet the system’s expectations” (Wilson, 2007). When performing a gesture incorrectly we could tell the user how to improve the hand posture for the gesture to be detected.

Another modification to the feedback in the hand contour regards the time when gestures are not being detected. In the third iteration we included a waiting time after the detection of any pre-stroke (1 second) and post-stroke (2 seconds). Before, in the second version of the prototype, we displayed the waiting time in a text at the bottom of the screen. For the third version of the prototype, we changed the waiting time from textual to visual, e.g. we filled the contour with red. Although we thought this would be a clear indication of ”stop”, the hand contour filled in red did not produce the behavior we looked for in the users. Even though we explained to the participants in the final evaluation that when the hand appeared in red no gestures would be detected, during the training phase some showed small signs of frustration when no post-strokes were detected in the waiting time and others showed some signs of confusion. It took some time for the participants to get used to this workflow. Maybe this was because they were too focused on finishing the gestures, or it may even have caused a visual overload. In any case, a better way to tell the user to wait could be found.

6.5.2 Feedback method

It is difficult to say if the type of feedback we use, e.g. visual, is the best for our input modality. Haptic feedback is not available in the M100 or Google Glass and before the final evaluation of the system we thought auditory feedback would cause an overload of information to the user. In retrospective, a good inclusion to the prototype would be to include auditory feedback and reduce the visual feedback. This could be a solution for the previous example of the waiting time. Instead of only showing the hand in red after pre-strokes and post-strokes, a ”beep” sound could be included as a more explicit confirmation of the classification. Then the users could relate better to the meaning of the hand in red, e.g. ”stop”. Even further, if we combine the previously suggested frame counter before a gesture is classified with the ”beep” sound after a classification, we would not need to include an explicit waiting time in our design.

7 Conclusion

In this thesis, we have argued that there is a need for new techniques to interact with eyewear computers. Voice commands, one-hand keyboards, and smartphones have been used to provide input but these have their own limitations, thus interaction with eyewear computers is still a challenge. We also argued that hand gestures can provide a natural and intuitive interaction with eyewear computers. We have presented a scenario were an eyewear computer assistant supports a surgeon during a typical ward round.

We have developed a prototype that supports hand-based gestures as input modality for eyewear computers. We have defined a set of gestures and developed a hand-based gesture detection method optimized for our set of gestures that relies on the characteristics of the hand. The prototype was developed in an iterative design approach with four iterations. During these iterations we held one brainstorming session, two formative evaluations, and the final evaluation of the system.

Our results show that our set of gestures are intuitive, most of them ergonomic, easy to remember and lack ambiguity. As a complementary measurement, the results show a precision of 96.91% and a recall of 89.52% of the gesture detection method.

We have suggested future improvements for our prototype. Moreover, as a result of our observations through the design of the prototype and the result of the evaluations, we have presented a set of design guidelines useful for future systems supporting hand-based gestures as input modality.

8 Acknowledgements

First, I would like to thank Thomas Pederson and Shahram Jalaliniya for their dedicated supervision throughout this project. Thanks to Diako Mardanbegi for his advice in Image Processing. Thanks to the PIT Lab members at ITU that participated in the evaluations. Thanks to my brother Diego for sharing his knowledge and advice in the writing process.

Furthermore, I express my deepest gratitude to my family and friends for their moral support and motivation through the entire process of writing this thesis.

References

ARGYROS, A. A. AND LOURAKIS, M. I. A. 2006. Vision-based interpretation of hand gestures for remote control of a computer mouse. In *Proceedings of the 2006 International Conference on Computer Vision in Human-Computer Interaction*. ECCV'06. Springer-Verlag.

BARDRAM, J. E., BALDUS, H., AND FAVELA, J. 2006. Pervasive computing in hospitals. In *Pervasive computing in Healthcare*. CRC Press.

BARDRAM, J. E. AND BOSSEN, C. 2005. Mobility work: The spatial dimension of collaboration at a hospital. *Comput. Supported Coop. Work 14,* 2 (Apr.).

BARRÉ, R., CHOJECKI, P., LEINER, U., MÜHLBACH, L., AND RUSCHIN, D. 2009. Touchless interaction-novel chances and challenges. In *Proceedings of the 13th International Conference on Human-Computer Interaction. Part II: Novel Interaction Methods and Techniques*. Springer-Verlag.

BENYON, D. 2010. Understanding. In *Designing Interactive Systems:A Comprehensive Guide to HCI and Interaction Design*. Addison-Wesley.

BRADSKI, G. AND KAEHLER, A. 2008. *Learning OpenCV*. O'Reilly Media.

CARLSSON, V., KLUG, T., ZIEGERT, T., AND ZINNEN, A. 2006. Wearable computers in clinical ward rounds. In *Applied Wearable Computing (IFAWC), 2006 3rd International Forum on*.

COLAÇO, A., KIRMANI, A., YANG, H. S., GONG, N.-W., SCHMANDT, C., AND GOYAL, V. K. 2013. Mime: Compact, low power 3d gesture sensing for interaction with head mounted displays. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*. UIST '13. ACM.

DAVIS, J. AND SHAH, M. 1994. Visual gesture recognition. *Vision, Image and Signal Processing, IEE Proceedings - 141,* 2 (Apr), 101–106.

DREWES, H. 2013. A lecture on fitts' law.

GANDY, M., STARNER, T., AUXIER, J., AND ASHBROOK, D. 2000. 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. IEEE Computer Society, Washington, DC, USA.

HALES, J., MARDANBEIGI, D., AND ROZADO, D. 2013. Interacting with objects in the environment by gaze and hand gestures. In *ECEM 2013 Proceedings*.

HAN, J. 2005. Classification and prediction. In *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc.

HARRISON, C., BENKO, H., AND WILSON, A. D. 2011. Omnitouch: Wearable multitouch interaction everywhere. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. UIST '11. ACM.

HARRISON, C., TAN, D., AND MORRIS, D. 2010. Skinput: Appropriating the body as an input surface. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '10. ACM.

HINCKLEY, K. 2007. Input technologies and techniques. In *Human-Computer Interaction Fundamentals*. CRC Press.

HONG, P., TURK, M., AND HUANG, T. 2000. Gesture modeling and recognition using finite state machines. In *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*. 410–415.

JALALINIYA, S., SMITH, J., SOUSA, M., BÜTHE, L., AND PEDERSON, T. 2013. Touch-less interaction with medical images using hand & foot gestures. In *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*. UbiComp '13 Adjunct. ACM.

JOSLIN, C., EL-SAWAH, A., CHEN, Q., AND GEORGANAS, N. 2005. Dynamic gesture recognition. In *Instrumentation and Measurement Technology Conference, 2005. IMTC 2005. Proceedings of the IEEE*. Vol. 3. IEEE, 1706–1711.

KIDD, C. D., ORR, R., ABOWD, G. D., ATKESON, C. G., ESSA, I. A., MACINTYRE, B., MYNATT, E. D., STARNER, T., AND NEWSTETTER, W. 1999. The aware home: A living laboratory for ubiquitous computing research. In *Proceedings of the Second International Workshop on Cooperative Buildings, Integrating Information, Organization, and Architecture*. CoBuild '99. Springer-Verlag.

KIM, J., HE, J., LYONS, K., AND STARNER, T. 2007. The gesture watch: A wireless contact-free gesture based wrist interface. In *Proceedings of the 2007 11th IEEE International Symposium on Wearable Computers*. ISWC '07. IEEE Computer Society.

KIM, J.-W., NAM, T.-J., AND PARK, T. 2014. Compositegesture: Creating custom gesture interfaces with multiple mobile or wearable devices. *International Journal on Interactive Design and Manufacturing (IJIDeM)*.

LEE, S. C., LI, B., AND STARNER, T. 2011. Airtouch: Synchronizing in-air hand gesture and on-body tactile feedback to augment mobile gesture interaction. In *Proceedings of the 2011 15th Annual International Symposium on Wearable Computers*. ISWC '11. IEEE Computer Society.

LYONS, K., BRASHEAR, H., WESTEYN, T., KIM, J., AND STARNER, T. 2007. Gart: The gesture and activity recognition toolkit. In *Human-Computer Interaction. HCI Intelligent Multimodal Interaction Environments*, J. Jacko, Ed. Lecture Notes in Computer Science, vol. 4552. Springer Berlin Heidelberg.

MISTRY, P., MAES, P., AND CHANG, L. 2009. Wuw - wear ur world: A wearable gestural interface. In *CHI '09 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '09. ACM.

MITRA, S. AND ACHARYA, T. 2007. Gesture recognition: A survey. *Trans. Sys. Man Cyber Part C 37,* 3 (May).

MOESLUND, T. B. 2012. *Introduction to Video and Image Processing*. Springer.

RAMAMOORTHY, A., VASWANI, N., CHAUDHURY, S., AND BANERJEE, S. 2003. Recognition of dynamic hand gestures. *Pattern Recognition 36,* 9, 2069–2081.

REKIMOTO, J. 2001. Gesturewrist and gesturepad: Unobtrusive wearable interaction devices. In *Proceedings of the 5th IEEE International Symposium on Wearable Computers*. ISWC '01. IEEE Computer Society.

SIEWIOREK, D. P. 2002. New frontiers of application design. *Commun. ACM 45,* 12.

SOLEM, J. E. 2012. *Programming Computer Vision with Python*. O'Reilly Media.

STARNER, T. 2001. The challenges of wearable computing: part 1. *IEEE Micro*.

STARNER, T., PENTLAND, A., AND WEAVER, J. 1998. Real-time american sign language recognition using desk and wearable computer based video. *IEEE Trans. Pattern Anal. Mach. Intell. 20,* 12 (Dec.).

SURAL, S., QIAN, G., AND PRAMANIK, S. 2002. Segmentation and histogram generation using the hsv color space for image retrieval. In *Image Processing. 2002. Proceedings. 2002 International Conference on*. Vol. 2.

TSUKADA, K. AND YASUMURA, M. 2002. Ubi-finger: Gesture input device for mobile use. In *Proceedings of APCHI*. APCHI '02.

WILSON, A. D. 2007. Sensor and recognition-based input for interaction. In *Human-Computer Interaction Fundamentals*. CRC Press.

YAMATO, J., OHYA, J., AND ISHII, K. 1992. Recognizing Human Action in Time-Sequential Images using Hidden Markov Model. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

YEASIN, M. AND CHAUDHURI, S. 2000. Visual understanding of dynamic hand gestures. *Pattern Recognition 33,* 11, 1805 – 1817.

Appendices

A Brainstorming Session

In a brainstorming session the participants are encouraged to react to each other comments and keep the discussion alive. This kind of discussion is good when little information about the topic is known (Benyon, 2010). This discussion helped in our gestures definition because we could gather information about what the people would be comfortable with when interacting with a system like the one we developed. Also, when a question is asked to an open group, the final answer will converge to what the group, or smaller groups, think and not only one person. That is why we decided to hold a brainstorming session at the beginning of the first iteration in order to find an intuitive set of gestures to interact with images. These actions were rotating and zooming an image, and switching images in a carousel fashion. We recruited 7 participants (age average=26 $\sigma = 2.15$, 3 female) for the session. 4 participants were students in Computer Science, 1 was a student in Multimedia Design, 1 was a student in Philosophy and 1 was a student in Chemical Engineering. None of the participants had experience with eyewear computers.

A.1 Method & Design

Brainstorming sessions require some stimuli like pictures or video to get the ideas flowing. That is why for this activity the group was shown a picture in a board and was asked to move the picture first by touching it and then by performing gestures in the air. The desired outcome for this study was to find a set of gestures for the user to interact with the images through smart glasses. We now describe the design of the session.

1. **Introduction.** We explained the nature of the session and the desirable outcome.

2. **First activity.** The participants debated and/or demonstrated how to perform actions to the picture by touching it. The actions were rotate, zoom, and swipe through pictures in a carousel fashion. This activity was done with the intention of making the users to feel comfortable and let loose. Also to have a baseline on the intuitiveness and ergonomics for the possible gestures.

3. **Second activity**. The participants debated and/or demonstrated how to perform actions to the picture **not** touching it, rather imagine that they can interact with it in the air. The actions were rotate, zoom, and swipe through pictures in a carousel fashion.

4. **Open questions.** Two open questions were asked in this part. Before the questions, the participants were explained how we envisioned the prototype for them to better understand the questions. The first question was: How would you start or stop the interaction with the smart glasses?. The second question was: How would you feel about learning a ”new language” or set of rules to interact with images?

A.2 Results

The outcome of the session was not as expected since no specific gestures for interacting with images where found or concluded. Even though the result was not as expected, other important information was gathered that facilitated the first definition of the gestures.

First, the participants were asked to rotate, zoom, and switch a printed picture placed on a white-board in front of them. The results can be seen in table 5.

Action	Suggested Gestures
Rotate	• Placed one hand (5 fingers pushing) at the center of the picture and rotated it towards the desired direction.
Zoom	• Grabbed picture in opposite corners (top-right and bottom-left) and enlarged or shrunk keeping the size proportions • Grabbed picture in opposite sides (right and left) to increase the width • Grabbed picture in opposite sides (top and bottom) to increase the height
Swipe	• Placed hand besides the picture and pushed it

Table 5: Results for the first activity of the brainstorming session. These are the most popular suggestions for each action while touching the picture.

In the second activity the participants were asked to perform the same actions but this time without touching the picture. This part did not throw the expected results since the participants could not agree in common gestures for some gestures. Regardless of not having the expected outcome we gathered important information about how the gestures should be. The only thing all participants agreed in this part of the activity was that gestures made with the palm of the hand pointing towards themselves are better.

In the third activity the users were asked how they would start or stop the interaction with the smart glasses and once again no concrete gestures were reached by this activity but important information was gathered. Under some uncertainty participants hinted that the hand could be placed in front of the camera with the palm towards the person itself. Another possible gesture was a splashing of the hand towards the glasses. The only thing the participants agreed in this question was that the gesture has to be something that one would not normally do.

To finalize this study the participants were asked their thoughts about learning a new (sign) language since the list of gestures can grow in a future. Most of the participants felt it was not a problem to learn gestures for a new way of interaction with the glasses as long as they are easy to remember or they are natural gestures.

Action	Suggested Gestures
Rotate	• Placed hand (5 fingers) towards user and rotated to desired direction • Placed 3 fingers pointing towards user and rotated to desired direction • Placed 2 fingers (L-shaped) and moved them to desired direction • Placed 1 finger towards user and made a rotate gesture (inverted U) towards desired direction
Zoom	• Pinch gesture with 2 fingers closing or opening the gap in between • Pinch gesture with 2 fingers using both hands and bringing hands closer or apart
Swipe	• Swipe movement using 1 or two fingers to the sides • Swipe movement using 1 finger up/down
Start or Stop	• Open hand with the palm towards user • ”Splashed” hand towards the user

Table 6: Results for the second activity of the brainstorming session. These are the most popular suggestions for each action without touching the picture

B First formative evaluation: Choosing the hand feedback.

The implementation of the first version of the prototype made us realize that there was an error in the design of the prototype. It was very difficult for the user to select anything on the lower part of the screen, almost 50% was un-clickable. In order to make the screen 100% clickable we decided to remove the live feed of the camera, display the background in black, and map the upper part of the screen to the entire screen. Using the first version of the prototype we realized that a very useful feature was being able to see the hand in the screen. This gave the user the possibility of modifying or adapting the hand posture to fit better with the gestures models. Thus, we knew that displaying the hand was an important feature.

B.1 Method & Design

With the background in black we faced the question: What kind of feedback of the hand would the user prefer? In order to answer that question we designed an informal evaluation to choose the hand feedback. We went around the IT University of Copenhagen and chose random students to be the participants for this evaluation. We recruited 9 people (2 female) to participate. 3 had experience with smart glasses. Three settings were presented to the users. Each setting had the task of clicking

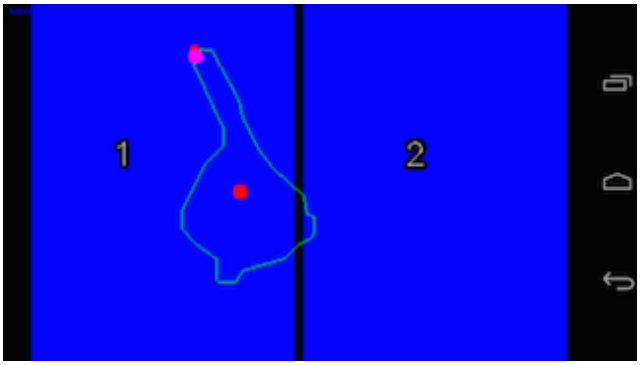


Figure 55: Result of the formative evaluation for choosing the hand feedback. Here, the hand contour is presented to the user in green and the live feed from the camera is blocked. This allowed us to map the upper half of the screen to the whole screen.

a number of rectangles appearing in the screen (see figure 55).

a) **Setting A.** The first setting shows the hand of the user and the live feed from the camera.

b) **Setting B.** The second setting shows the hand contour in green without the live feed from the camera.

c) **Setting C.** The third setting shows dots placed on the tip of the fingers without the live feed from the camera.

After the participants completed the task on each setting, they were asked to answer a questionnaire with three simple questions regarding their preferences.

1. Which setting was easier to use?
2. Which setting was more convenient?

3. Which setting was more accurate? (when clicking)

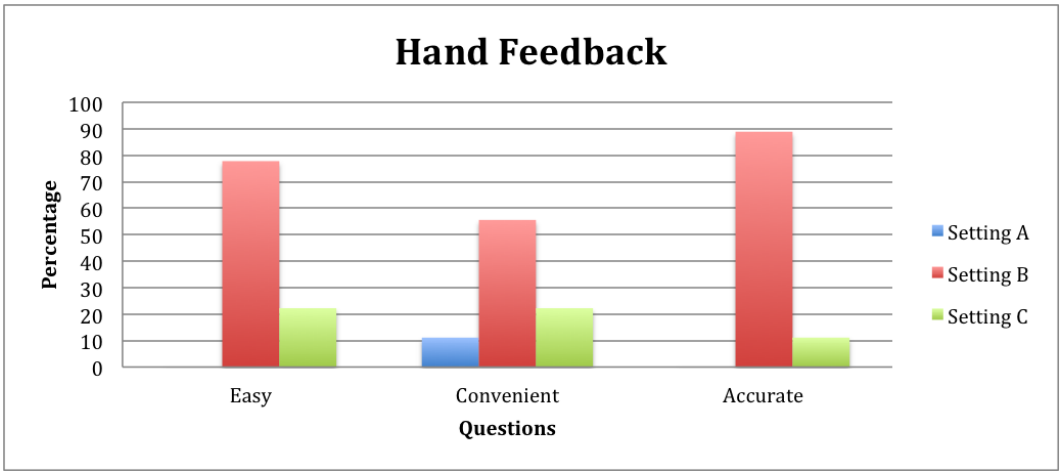


Figure 56: Second User experiment.

Even though our solution for the un-clickable lower part of the screen was to block the camera feed and display the background in black (settings B and C), we needed to make sure that this setting was preferred over the initial set up (setting A).

B.2 Results

The results can be seen on figure 56. It is clear that the participants preferred setting B in every aspect. For the first question 7 out of 9 participants (77%) chose setting B as the easiest setting to use. In the second question 5 out of 9 participants (55%) chose setting B as the more convenient. And in the third question 8 out of 9 participants (88%) chose setting B as the setting that allowed more accurate clicking.

C Second formative evaluation: First exposure of the prototype

This formative evaluation was held to evaluate the second version of the prototype. Here, our main goal was to analyze the performance of *PointSelect*, the performance of the gesture detection method, and the interaction with the prototype. Before this evaluation the only users that had truly tested the system were ourselves. We had little information about how other users perceived the system. This evaluation confirmed we were going in the right direction. 6 participants (age average=27 $\sigma = 3$, 3 female) were recruited to participate. Half of the participants had intermediate computer skills, and the other half were expert computer users. One participant had myopia, and one participant had a disability in the ring finger. This disability did not interfere with the evaluation since the ring finger is not used for any of the evaluated gestures. All of the participants were right-handed.

It is important to mention that the evaluation was meant to be done with the M100 Smart Glasses, but during the session we encountered problems. The smart glasses were heating after several minutes of use causing them to power off and the battery seemed to be draining out. Thus, we carried on the evaluation with a Nexus 5 smartphone recreating, to some extent, the eyewear computer set up by holding up the phone in front of the participants. At the end of the session the participants could try the application in the M100 Smart glasses in an informal way. We address these challenges in the next chapter. In the rest of the section we first present the design of the experiment. Then, we follow with the results.

C.1 Design

The evaluation was designed to take place in two phases. The first phase was an experiment with *PointSelect* and the second involved a scenario that had to be followed by the participants.

C.1.1 *PointSelect* experiment

The goal of this experiment was to measure the following three aspects of the gesture.

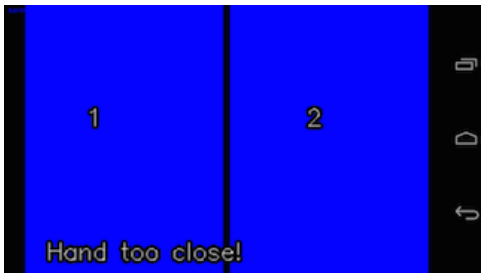
1. Performance of the *pointSelect* gesture regarding the participants, not the system. We measured the performance with the time it took the participant to finish each condition.

2. Best clickable area. We measured the easiest area to click according to the accuracy on each area of the screen.

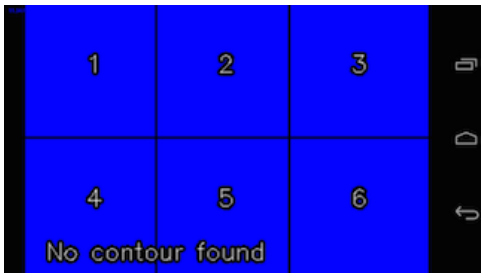
3. Smallest size for clickable objects. We measured the smallest size for clickable objects by comparing the errors in each condition.

For this we designed a small application with 3 conditions. The goal for the participant was to click all the buttons in the screen as fast and precise as possible in the correct order. The first condition presented the base line with two buttons. The second condition contained 6 buttons, and the third condition contained 12 buttons. Thus, each time a participant advanced to the next condition one row and one columns of buttons was added. At the beginning of the condition all the buttons on screen appeared in blue color and as the participant clicked the buttons they would turn green. During this experiment the only gesture available was *pointSelect*.

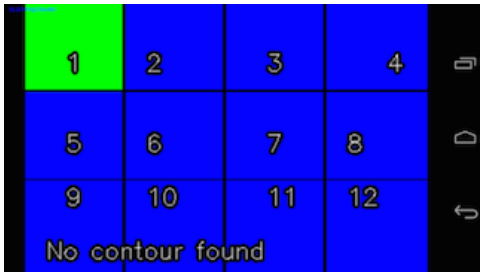
The experiment was carried out the following way. First the participant received an explanation of the experiment, followed by a description of the gesture. To let the participants get used to the application, we asked them to complete the task as part of a training phase. Finally, the participants went through the conditions in a formal way.



(a) Condition 1. Baseline.



(b) Condition 2.



(c) Condition 3.

Figure 57: Conditions for the *PointSelect* experiment.

C.1.2 Scenario

The second phase of the experiment assessed the user experience and usability of the prototype. First, the users were explained that they were going to follow a scenario. Next, we explained the gestures to be used and their actions. Then, we provided an example of how to complete the scenario by doing it ourselves while they were watching the screen. The participants also went through the scenario as a training phase. Finally, they went through the scenario in a formal way. The scenario in the experiment is based on the ward round scenario presented in chapter 1. The actions in the scenario to perform by the participants are described next.

1. Select patient Tom.
2. Read the first sentence on screen. After that sentence there are some more instructions. Please follow the instructions. (Select image 3)
3. Rotate the selected image upside-down.
4. Zoom in the image twice.
5. Change the center of focus for the image.
6. Zoom out the image twice.
7. Go back to the patient record.

The previous actions include all the dynamic gestures in our vocabulary. After completing the scenario the participants were asked to answer a questionnaire. The complete questionnaire can be seen in Appendix D. The questionnaire had three parts. In the first part we asked the participants

to identify the gestures in a table by matching the corresponding name with the gesture's picture. In the second part we asked three questions 1) How intuitive are the gestures? 2) How easy are the gestures to perform 3) How easy are the gestures to remember?. We omit *Init* from the identification of the gestures since the scenario was not designed including a 40 seconds waiting period.

C.2 Results

As previously mentioned, we started the evaluation with the M100 smart glasses but we encountered problems with heat and battery, and we carried on with the evaluation in a Nexus 5 smartphone. The previous modification caused the loss of data for the first two participants. Thus, the results presented here are from the last 4 participants.

C.2.1 *PointSelect* experiment results

Performance. In order to measure the performance of the participants we averaged the time of completion (figure 58). For condition 1 the average in seconds was 12 ($\sigma = 6.98$), for condition 2 was 31.75 ($\sigma = 9.91$), and for condition 3 was 65.5 ($\sigma = 15.33$).

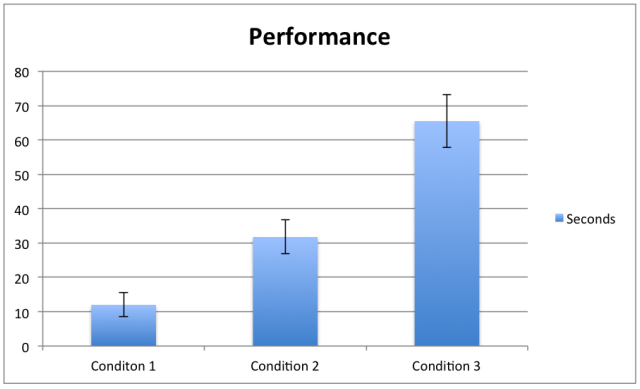
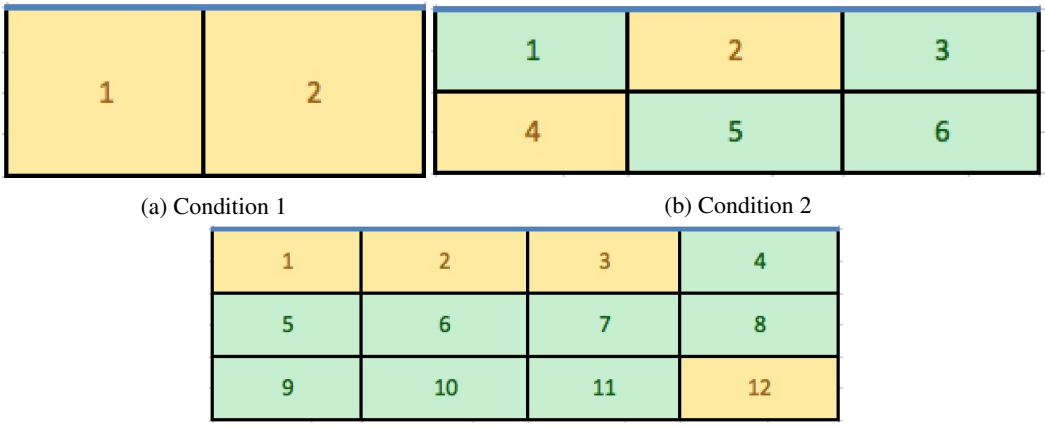


Figure 58: Performance of participants. Average seconds for the participants to finish the conditions.

Best clickable area. The easiest area to click was measured with the accuracy of the participants. A click was considered incorrect when a square other than the one in turn was clicked. In figure 59 we can see the sum of all the wrong clicks for each square. A green square means that no wrong clicks were done in that turn, e.g. all participants clicked square 3 in condition 2 in the first try. A yellow square means that 1-3 wrong clicks were done in that turn. The lowest amount of wrong clicks we encountered was 3, less than 1 in average per participant.

Smallest clickable objects. To measure the size of the smallest clickable object, or square in this case, we look at the number of wrong clicks in a different way. Instead of counting the incorrect clicks per square, we average them per condition. The average number of incorrect clicks in condition 1 was 0.75 ($\sigma = 0.70$). The average for condition 2 was 0.50 ($\sigma = 0.51$), and for condition 3 was 1.5 ($\sigma = 0.90$). (figure 60)

Analysis. The time to complete each condition seems to have a direct correlation with the number of targets. The time for condition 2 is 3 times bigger than condition 1 (6 targets vs 2



(a) Condition 1

(b) Condition 2

(c) Condition 3

Figure 59: Accuracy of participants. Sum of wrong clicks in each condition. The green squares mean no participant clicked the wrong square. The yellow squares mean 1-3 wrong clicks were done when trying to click that square.

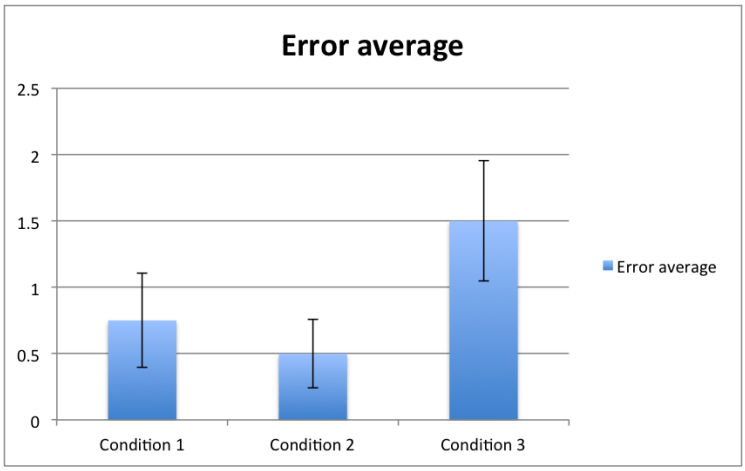


Figure 60: Average of errors in each condition.

targets), and the time for condition 3 is 2 times bigger than condition 2 (12 targets vs 6 targets). But, because the data set is very small, we can not make any conclusions.

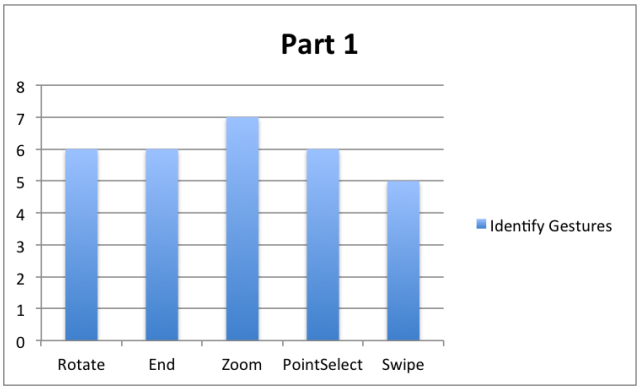
The accuracy for each square does not reflect a clear definition of the best clickable area. Instead, we get a surprising result in the first condition, which was supposed to be baseline for the rest of the conditions. If we only consider condition 3, we could say that the middle of the screen is the best and easiest clickable area. But this space was also available for the participants in the previous conditions and still incorrect clicks were made. As with the performance, we can not make any conclusions for this aspect. In retrospective, a fourth condition could have been a good option for this particular aspect, but at the time we felt the squares were too small.

The incorrect click average per condition also gives us a surprising result with condition 1 having

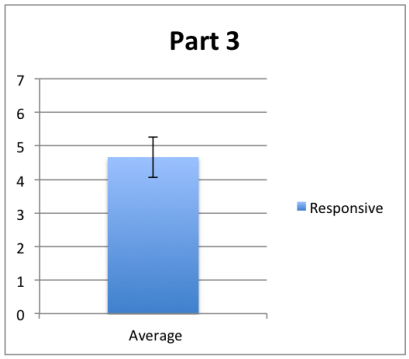
a higher average than condition 2. Consequently, we can not make any assumptions for the smallest clickable size.

C.2.2 Questionnaire results

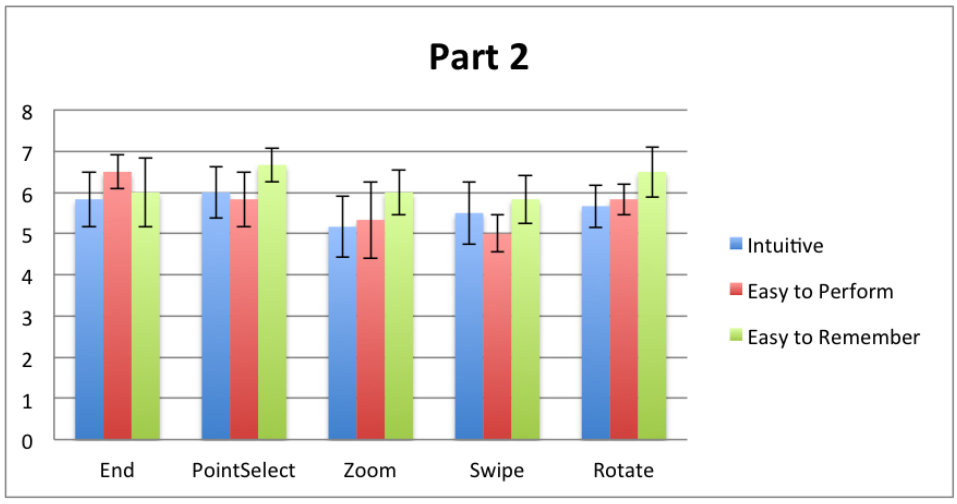
The problems with the M100 Smart glasses did not interfere with the questionnaire. The results presented here are from the 6 participants.



(a) Results for first part.



(b) Results for third part.



(c) Results for second part.

Figure 61: Questionnaire results for first evaluation of the system.

First part. For the first part of the questionnaire we asked the participants to identify the gestures (figure 61a). The results show that only one participant confused *Zoom* with *Swipe* and chose *Zoom* twice, with the rest of the gestures being correctly identified.

Second part. For the second part of the questionnaire we asked the participants about 1) the intuitiveness of the gestures 2) easiness to perform the gestures and 3) easiness to remember the gestures (figure 61a). Table 7 summarizes the results. We can see that *Zoom* and *Swipe* have the lowest scores on each question.

Gesture	Intuitive	Easy to Perform	Easy to Remember
End	$\bar{x} = 5.8, \sigma = 1.3$	$\bar{x} = 6.5, \sigma = 0.8$	$\bar{x} = 6.0, \sigma = 1.7$
PointSelect	$\bar{x} = 6.0, \sigma = 1.3$	$\bar{x} = 5.8, \sigma = 1.3$	$\bar{x} = 6.6, \sigma = 0.8$
Zoom	$\bar{x} = 5.2, \sigma = 1.5$	$\bar{x} = 5.3, \sigma = 1.8$	$\bar{x} = 6.0, \sigma = 1.0$
Swipe	$\bar{x} = 5.5, \sigma = 1.5$	$\bar{x} = 5.0, \sigma = 0.8$	$\bar{x} = 5.8, \sigma = 1.1$
Rotate	$\bar{x} = 5.7, \sigma = 1.0$	$\bar{x} = 5.8, \sigma = 0.7$	$\bar{x} = 6.5, \sigma = 1.2$

Table 7: Results for second part of the questionnaire.

Third part. For the third part of the questionnaire we asked the participants about the responsiveness of the system (figure 61b). We consider the third part to be our worst result with an average of 4.6 ($\sigma = 1.21$).

Analysis. The results for the questionnaire are good for the first two parts. This tells us that the definition of the gestures follows our metrics of *intuitiveness, ergonomics, lack of ambiguity and ease of recognition.* The first part shows that the gestures are *easy to recognize and they lack ambiguity*. The second shows that the gestures are *intuitive*, easy to perform (*ergonomic*) , and easy to remember. The third part tells us that the responsiveness of the system is poor. We believed this was caused by the lack of user feedback.

	Classified Gesture					
Actual Gesture	PointSelect	Swipe	Rotate	Zoom	End	Idle
PointSelect	38					5
Swipe	4	10				
Rotate	8		20			1
Zoom	3			27		6
End					10	
Idle						

Figure 62: Confusion Matrix for the gesture detection in the scenario.

C.2.3 Gesture detection performance

The classification of the gestures during the second phase of the evaluation is summarized in a confusion matrix 62 which helps us calculate the precision and recall of the classification. Precision (equation 3) refers to the fraction of the classifications that are correct, and recall (equation 4) refers to the number of classifications relative to the number of gestures performed (Han, 2005).

$$P = \frac{truepositives}{truepositive + falsepositives} \tag{3}$$

$$R = \frac{truepositives}{truepositive + falsenegatives} \tag{4}$$

The performance of the gesture detection method also improved with a 87.50% precision and a 89.74% recall. The confusion matrix shows a lot of false positives classifying *PointSelect*'s pre-stroke, which indicated to us the existence of a bug in the classification of the gestures. This also caused some frustration to the participants. Also, it seems the prototype remained idle in several occasions. This may also be causing the users to think the prototype is not responsive.

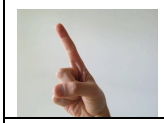
D Questionnaire for the Evaluations

General Information

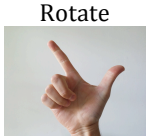
1. Age: _____
2. Gender
   a. M ___
   b. F ___
3. Do you have any previous experience with smart glasses?
   a. Yes ___
   b. No ___
4. What is your experience with computers?
   a. Beginner ___
   b. Intermediate ___
   c. Expert ___
5. Do you have any form of eye, hand, or arm disability?
   a. No ___
   b. Yes ___. Please specify: _____
6. Are you:
   a. Left handed: ____
   b. Right handed: ____

Questionnaire

1. **Can you identify the gestures?**
   Mark the box corresponding to the picture on the left.

Gesture	Rotate	End	Zoom	PointSelect	Swipe
					
					
					
					
					

2. Pleas answer the following questions

Gesture	How intuitive is the gesture?	How easy is the gesture to perform?	How easy is the gesture to remember?
End 	1 2 3 4 5 6 7 1 = not intuitive 7 = very intuitive	1 2 3 4 5 6 7 1 = not easy 7 = very easy	1 2 3 4 5 6 7 1 = not easy 7 = very easy
PointSelect 	1 2 3 4 5 6 7 1 = not intuitive 7 = very intuitive	1 2 3 4 5 6 7 1 = not easy 7 = very easy	1 2 3 4 5 6 7 1 = not easy 7 = very easy
Swipe 	1 2 3 4 5 6 7 1 = not intuitive 7 = very intuitive	1 2 3 4 5 6 7 1 = not easy 7 = very easy	1 2 3 4 5 6 7 1 = not easy 7 = very easy
Zoom 	1 2 3 4 5 6 7 1 = not intuitive 7 = very intuitive	1 2 3 4 5 6 7 1 = not easy 7 = very easy	1 2 3 4 5 6 7 1 = not easy 7 = very easy
Rotate 	1 2 3 4 5 6 7 1 = not intuitive 7 = very intuitive	1 2 3 4 5 6 7 1 = not easy 7 = very easy	1 2 3 4 5 6 7 1 = not easy 7 = very easy

3. How responsive do you find the system?

1 2 3 4 5 6 7

1 = not responsive 7 = very responsive

4. Additional comments:

_____

_____

_____

Thank You!