Master's Thesis

User-independent recognition of dynamic hand gestures using a dataglove

Ing. Mario S. Ganzeboom
s0134848

October 28, 2010

Graduation Committee:
Dr. P.E. (Paul) van der Vet
Dr. M. (Mannes) Poel
Dr. Ir. F.W. (Wim) Fikkert

Human Media Interaction Group
Faculty of Electrical Engineering, Mathematics and Computer Science
University of Twente, Enschede
The Netherlands

'An idea. Resilient, highly contagious. Once an idea has taken hold of the brain it's almost impossible to eradicate. An idea that is fully formed - fully understood - that sticks; right in there somewhere.'

(Character of Cobb, From the film 'Inception', 2010)

'A picture is worth a thousand words. An interface is worth a thousand pictures.'

(Ben Shneiderman)

'We have to learn to see technical devices and software as others see them. Feeling a sense of pride in getting a device to do what we want doesn't mean we're clever, it means the design was wrong.'

(Jeremy Allison)

Summary

The goal of this thesis is the user-independent recognition of dynamic, single-handed gestures using a dataglove. This research focuses on the technology behind the recognition of these gestures. It wants to try and find suitable technologies, compare them and research ways to optimise the use of these technologies to increase recognition performance.

The dynamic hand gestures described in this research were defined by researching the tasks users do when controlling a large display surface. Nine gestures were defined to cover different aspects of navigating a user interface. Using a dataglove for hand pose data and an additional sensor for position and orientation data, data was recorded on all nine gestures from eight users in a Wizard of Oz experiment. The aim of this experiment was the simulation of a real-world environment in which users control two applications with the nine gestures.

With the recorded data three types of recognisers that were suitable for the goal of this thesis were trained and tested: Discrete Hidden Markov Models (DHMMs), Continuous Hidden Markov Models (CHMMs) and Latent-Dynamic Conditional Random Fields (LDCRFs). Different data pre-processing methods and parameter tuning were applied to try and increase their recognition rate. Contrary to expectation, the DHMMs and CHMMs performed poorly and only LDCRFs were able to achieve a satisfying recognition rate on the recorded data set. Initial testing of the LDCRFs on user independence also showed satisfying recognition rates.

This thesis shows that the user-independent recognition of dynamic hand gestures using a dataglove is feasible. Even though this research describes only a part of the process involved in designing hand gesture interaction, it can without a doubt play an effective role in its realisation.

Samenvatting

Het doel van deze masterscriptie is het gebruikersonafhankelijk herkennen van dynamische handgebaren met behulp van een dataglove. Het onderzoek richt zich op de technologie achter het herkennen van deze gebaren. Er is gezocht naar geschikte technologieën en deze zijn met elkaar vergeleken. Daarnaast zijn manieren onderzocht om ze te optimaliseren met als doel het herkenningspercentage te verhogen.

Negen dynamische handgebaren worden in deze scriptie beschreven en zijn gedefinieerd aan de hand van een analyse van de taken die gebruikers uitvoeren wanneer zij een groot scherm bedienen. In een zogeheten 'Wizard of Oz'-experiment is met behulp van een dataglove en een extra sensor data opgenomen met betrekking tot de vorm, positie en oriëntatie van de hand. In dit experiment werd ernaar gestreefd een zo echt mogelijke omgeving na te bootsen waarin de gebruikers twee applicaties bedienen met de negen gebaren.

Vervolgens zijn drie verschillende recognisers getraind en getest: Discrete Hidden Markov Models (DHMMs), Continuous Hidden Markov Models (CHMMs) en Latent-Dynamic Conditional Random Fields (LDCRFs). Om het herkenningspercentage van deze recognisers te verhogen, zijn verschillende datavoorbewerkingsmethoden en de effecten van het afstellen van de recogniserparameters onderzocht. Geheel onverwachts presteerden de DHMMs en CHMMs slecht, de LDCRFs waren de enige met een bevredigend herkenningspercentage. De LDCRFs gaven tevens bevredigende resultaten in de initiële gebruikersonafhankelijkheidstests.

Dit onderzoek toont aan dat het gestelde doel haalbaar is. Hoewel dit onderzoek slechts een deel beschrijft van het ontwerpproces van handgebareninteractie, kan het zonder twijfel een effectieve rol spelen in het realiseren ervan.

Preface

Since my teenage years I have been fascinated by the interfaces used in the television series 'Star Trek: The Next Generation'. The speech interface in particular fascinated me and I daydreamed more than once about its application in future homes. It enabled control and the exchange of knowledge by the intuitive use of verbal language or so it seemed. At the beginning of my master's I was introduced to the Natural Interaction paradigm and learned that speech interfaces are designed with naturalness in mind to be intuitive to use. This introduction fed my interest in natural interaction and its use of the vision, speech and touch modalities. After doing an independent research assignment involving the gestural modality I knew I wanted to do a larger project to further explore the possibilities of gesturing. This became my final research project for obtaining my master's degree. What lies before you are its results.

Even though I believe in the 'User-Centered Design' philosophy, in which interfaces are designed from a user perspective, I wanted to approach this project from a technological perspective. This allowed me to explore the technologies behind the recognition of dynamic hand gestures, to research how these technologies perform and to try to find ways to optimise their use.

To quote one of my supervisors at the beginning of the project: 'Data is everything with these technologies'. A statement which is inherently true when considering that the machine learning techniques explored in this research solely use data to 'learn' pattern characteristics. I personally have found it to be true after having observed the positive and negative effects various data preprocessing methods have on gesture recognition.

As every data set is different and previous research did not provide much detail, I tried to get a feel for the data characteristics by iteratively training and testing

recognisers. Although this gave me the insight I wanted, I sometimes made one iteration too many, which jeopardised my schedule. Fortunately, my supervisors were there to drop me a hint as to when to 'terminate' the iterative loop.

This thesis mainly contributes to the field of gesture recognition by showing that user-independent recognition of hand gestures is feasible using a technology that to the best of my knowledge had previously not been used for dynamic hand gesture recognition as presented in this thesis. A comparison is also made with commonly used technologies. In addition, contributions are made on the topic of data pre-processing in support of gesture recognition. Some research is also done on the numerical stability of the algorithm implementations used.

User-independent recognition of dynamic hand gestures remains a challenging task. I hope this thesis provides inspiration for future research in the field of hand gesture interaction.

Acknowledgements

In the course of doing the research for and writing my master's thesis I received support from many people. Here I would like to thank them. First and foremost my supervisors, Paul van der Vet, Wim Fikkert and Mannes Poel who have supported me all the way. Thank you Paul for always letting me see my research as being part of a bigger picture and for encouraging me to find out why I found the results I did. Thank you Wim, most of all for showing me different angles from which to look at my research and for putting it into perspective. And thank you Mannes for your insight and comments on the recogniser technologies. I would also like to thank Han Rauwerda from the University of Amsterdam for showing me the setup of the e-BioLab and for discussing with me the tasks users do. I also owe a word of thanks to Natalie Ganzeboom, who reviewed large parts of my thesis and to my fellow graduates for providing a productive atmosphere in the 'graduate room'. Finally, I would like to thank my family and friends for always supporting and listening to me when I just needed to talk.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

During the 1980s personal computers were introduced into working environments with the office as the most striking example. Interaction was provided through a keyboard, mouse and a relatively small display. In the last few decades computers evolved and displays became larger. In addition, computers and displays are no longer restricted to working environments. Nowadays they can be found in shopping centres, train stations, municipal offices and other public environments, in which large display surfaces often provide context-related information in an interactive way. Interacting with these surfaces by using a keyboard and mouse can be obtrusive and constraining in such environments [86, 10], which has serious consequences for how users experience these environments. The interaction design that is part of these environments should become more natural and intuitive to the user [32] and it should not draw attention to the technology behind it or impose a high cognitive load on the user. Moreover, the interaction with computers (or machines in general) should be adapted to the user and not force the user to adapt [50]. This change in the way of thinking about Human Computer Interaction has resulted in novel ways of interaction.

The different human modalities inspired scientists to design new ways of natural interaction. Speech (one of the modalities) recognition technologies have existed since the early 1950s [14], but it was not until the mid 1980s when advances in statistical processing sparked a renewed interest in the field.

Another human modality, the modality of touch inspired touch and multi-touch surfaces for computer systems, which were invented in the early 1980s. They redefined interaction through direct manipulation, which allows the user to manipulate objects directly using actions that (loosely) correspond to those in the

physical world [76, 30]. Advances in technology increased the size of newer generations of touch and multi-touch surfaces while greatly reducing their cost. As a result the use of touch and multi-touch surfaces in different environments is increasing steadily. Mouse actions were simply replaced by touch actions, with which a user navigated a graphic user interface [26]. More efficient recognition techniques popularised the recognition of strokes and led to the recognition of single-handed strokes of different forms [42]. In the last decade, advances in multi-touch technologies increased the popularity of the recognition of bimanual gestures on touch surfaces [18, 27].

However, research into gesture-based interaction was not limited to touch surfaces. From the mid 1980s research has also been done on interaction through gestures with parts of the body that are interpreted by a computer [99, 59]. Gesture interaction is a broad field and includes the recognition of full body motion [62], as well as head motion [56], facial expressions [63] and hand gestures [29], the latter being the subject of this thesis. A user interacting through hand gestures makes gestures with the hands and arms in mid-air. These gestures are observed and interpreted by a recogniser system, different types of which have been researched. Most of these systems can be categorised into one of two groups, based on either static hand pose recognition or dynamic gesture recognition [86, 54]. The systems in the former group measure and interpret the actual, static shape of the hand. A good example of this is the research on the recognition of sign language [37]. Dynamic gesture recognition on the other hand involves gestures that change over time such as the waving gesture [11]. While making this gesture, the position and orientation of the arm and hand vary over time, the specific variations of which need to be recognised, which is precisely what makes this type of recognition dynamic. Dynamic gesture recognition makes the recognition of gestures that are less constrained possible, which gives the user more freedom [79].

Gesture recognition can be further categorised into user-dependent [48] and user-independent [70] recognition. The performance of user-dependent recognisers depends on them being tuned to the individual user. They therefore require some type of calibration in order to become familiar with a particular user's specific characteristics before they can be used effectively [60]. User-independent recognisers do not needthis type of calibration and instead use their capacity to generalise the gestures of users: any user can use this type of recogniser immediately as no preparation (e.g. calibration) is required.

The focus of this thesis is on the development and evaluation of a dynamic, user-independent gesture recogniser. The following sections give a clear overview of the goals and boundaries of this research.

1.1. Motivation

A technology-driven approach was chosen due to the availability of certain hardware: a dataglove consisting of seventeen functional flex sensors. In previous research this glove has been used to measure the pose of a hand and provide data to a pose recogniser [23]. Three poses were defined to recognise four interface actions: clicking, dragging, zooming and rotating. A fourth pose was defined to function as a rest pose, to which no interface action was assigned. A training set of fifty samples for every pose was created. The test set consisted of fifteen samples in total that were different to those in the training set. All test samples were from a single person. The test results showed that fourteen out of the fifteen (93%) test samples were recognised correctly if the user's hand returned to the resting pose in between gestures. All fifteen (100%) were recognised correctly when the hand transitioned smoothly from one pose to the next. No significant tests were done with other users as part of this research. However, tests with another user in an informal setting showed that the recogniser only randomly recognised the correct poses at best. The recogniser was only trained and tuned to a single specific user.

To embrace the unobtrusiveness natural interfaces strive for, the interaction should not draw attention to the technology, but rather to the task at hand. For both user-dependent and user-independent recognisers learning has to take place to memorize the gesture set [38]. However, users of user-independent recognisers do not have to go through a calibration or training phase prior to using them.

The research described at the beginning of this section concerns hand pose recognition, which is particularly relevant for sign languages. Gestures that change over time, also known in the field as 'dynamic', are more relevant to other contexts. They are often used to recognise human motion [94], thereby diversifying possible gesture interaction. This is also why in this thesis the choice was made to research user-independent recognition of dynamic gestures.

1.2. Thesis subject

Although the field of gesture recognition includes many ways of gesturing this thesis is about recognising hand gestures. It was inspired by the results from and owes a word of thanks to Fikkert [20], who describes single-handed and bimanual gestures for controlling large display surfaces that are out of arms-reach. Fikkert's main focus is to find suitable gestures while taking into account human behaviour. The focus of this thesis is the technology behind the recognition of these gestures in order to enable recognition and research ways to optimise the use of those technologies to increase recognition performance. This thesis focuses on single-handed gestures as their recognition is less complex.

The drop in recognition performance in previous research [23] means the gesture recognition process cannot be extended to other users. Furthermore, dynamic gestures are extremely suitable for controlling large displays based on the fact that the dynamic gestures that Fikkert [20, ch. 4 & 5] considers all contain some form of motion in either arm, hand or fingers. This thesis researches the challenges posed by user-independent recognition of dynamic gestures when applied to the control of large display surfaces that are out of arms-reach.

One of the challenges is to define a gesture set for users to control large display surfaces. For example, several gestures can be defined for controlling a display for zooming in on an image. In front of a large display you could make a pulling gesture, one you make when pulling a closed door towards you to open it. Similar to pulling the door towards you, pulling the image towards yourself allows a closer look. Another 'zoom in' gesture could start with both hands put together. Moving the hands in opposite directions could then start the zooming process. A third gesture for zooming in could involve moving the tips of the thumb and index finger from or to each other, while keeping the remaining three fingers in a fist. This is the so-called 'Pinch' gesture [91]. From the user's perspective the recognition of all of these gestures would be ideal. However, when seen from a technological perspective the recognition of all gestures becomes very complex. As this thesis focuses on a solution to hand gesture recognition from a technological point of view this complexity needs to be limited. That way the necessary calculations remain computationally tractable. As a consequence a small subset of hand gestures needs to be defined, which is done according to the existing literature on hand gesture recognition and the context of this thesis.

Another challenge is the usability of the interface, which depends among other things on the percentage of successfully recognised gestures. For instance can

users complete their tasks successfully if there is a margin for error or is there little to no room for error? Qualitative empirical research involving users is required to answer this question, which is not part of this research due to its focus on recognition technology. This research therefore assumes a linear relation between usability and percentage of successfully recognised gestures.

Its goal is the user-independent recognition of dynamic gestures, which in the most ideal situation means that every user regardless of age, gender, size, etc., should be able to simply put on the glove and start operating the interface without calibration or user training. This leads to many variables and a very complex system. In order to make the research slightly less complex and to be able to clearly see how different technologies and their parameters affect the recognition percentage, a particular group of users is targeted in this thesis, the study of which will reveal any characteristics that need to be taken into account during the recognition process.

The recognition technology that is used greatly influences the performance of the recognition interface. The choice of technology depends on what kind of gesture is selected for controlling large display surfaces. As the purpose of this thesis is to recognise dynamic gestures, a recognition technology that takes temporal changes into account is the better choice. When a selection of recognition technologies has been made, experimenting with different parameter settings is also a possibility to further increase the percentage of successfully recognised gestures [19, 31]. The training data used to train the recogniser is also of great importance, because when noisy data is used, the interface has more difficulty recognising the gestures. Moreover, the training data should not be limited to one user, but should include at least a few users, a representative sample of the target group. This way the gesture recogniser increases its capacity to recognise gestures from users not available during training [93], which raises the following question: 'How does the number of users available at training affect this capacity?'

In short, having taken on the challenges described in the previous paragraphs, the ultimate goal of this thesis is to achieve a percentage of successfully recognised gestures that is usable for a large group of users.

1.3. Research questions

This research would like to answer the following questions with respect to the above-mentioned goal.

1. Can dynamic hand gestures be recognised independent of users?
   a) Which technologies are suitable for this purpose?
      i. What effect does pre-processing the data set have on the performance?
      ii. What effect does the tuning of parameters have on the performance?
      iii. How does a traditional technology compare to a novel one?
      iv. What effect does the quantity of available user data have on the performance?
   b) Can a recogniser be made user independent for a large group?

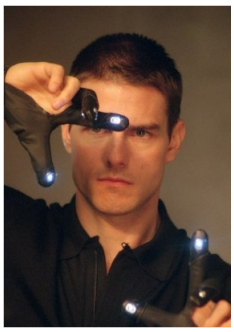
1.4. Research context

In the film 'Minority Report'[1], the main character, John, is a chief of police who tries to prevent crimes before they take place. John and his team have access to a special crime lab. In this lab John stands in front of a transparent display and sifts through large amounts of data (mostly images and video's) about the future. One of the methods he uses to control the interface is hand gesture interaction out of arms-reach of the display. John wears special gloves and makes single-handed or bimanual gestures to control the interface. Figure 1.1a and 1.1b show John's gloves and John controlling the interface.

Although this example may seem futuristic and unfeasible, it has been done. The scientists and developers who helped shape a realistic image of the technologies shown in the film, were able to make a hand gesture interface with similar features to that in the film. It was and still is being developed by Oblong Industries[2] and their spatial operating environment as they call it has been christened 'g-speak'.

John's fictional crime lab is a good example of the use of large displays, which are also used in the real world. As Fikkert [20] mentions, large displays can also be found in future homes [84], offices, schools and other public environments. In

[1] See http://www.imdb.com/title/tt0181689/. Last visited on July 13^{th}, 2010.
[2] See http://www.oblong.com. Last visited on July, 13^{th}, 2010.



(a) *Special gloves John uses to make gestures.*

(b) *John controlling the interface and reflecting with team members.*

Figure 1.1.: *Stills from the film Minority Report.*

future homes they will be the interface for the interactive systems in the home. In offices they will be used to display and interact with images and documents during meetings. Schools will use them to display animated learning aids to help students with their studies. Municipal offices, examples of public environments, will use large displays to provide information to visitors, enabling them to find their way around the building.

A more creative use of large displays in public environments is interactive art for example. Surrounding users with display surfaces would immerse them in a virtual world, as Fikkert [20] mentions, which they could navigate and in which they could manipulate objects [17]. This world can either be a realistic or virtual projection. It is expected that users explore these worlds independently and find out about its functionalities. They could also be artistic canvasses. The goal is to let the users be creative and make their own artistic creations. Museums are also good examples of public environments which could benefit from gesture interaction [39].

Fikkert also mentions operating theatres as locales for gesture-based interfaces. Surgeons could access patient information, for example, MRI, CT and X-ray images themselves without having to ask support staff to access it for them. In addition, gesture-based interfaces have the advantage that surgeons do not have to touch any form of controller to access patient information, which keeps their hands sterile and allows the surgeon to remain at the patient's side.

Busy shopping streets are examples of environments we are all familiar with. Shop keepers increasingly use displays in their windows to advertise the products to the public. The displays do not only show information about the shop or

the products in the window, but passers-by can also interact with them [89]. Displays that are able to detect when passers-by stand in front of them can adapt the information displayed. Passers-by can also interact personally by browsing the products available in the shop using simple taps, knocks or gestures. The interaction does not have to be limited to the detection of someone standing in front of the display, as mobile phones could take this to the next level by exchanging information with the display.

Life scientists have to deal with large amounts of experimental data [15]. To fully address this data, a multidisciplinary team of experts doing complex and computationally intensive analysis and experimentation is needed. In an effort to accelerate the process of analysing and visualising large biological data sets, discuss results and address biological research questions, an e-BioLab was built at the University of Amsterdam [69]. The e-BioLab provides the life scientists with a large, high-resolution, tiled display, high-bandwidth connectivity and access to high-performance computing. A multidisciplinary team of scientists working together co-located trying to solve problems in fields of biology, genetics, bioinformatics and micro array experiments amongst others. The environment employs a methodology facilitating the re-use of methods and applies domain dedicated Problem Solving Environments. The use of the e-BioLab in actual practice has turned out to facilitate lively and focused discussions. Insights into problems were gained at a faster pace, which would have been nearly impossible without the use of the e-BioLab. The large display surface used in the e-BioLab is a tiled display that consists of twenty 21-inch LCD computer screens arranged in a 4 by 5 matrix. The software used in the e-BioLab to render the display is the Scalable Adaptable Graphics Environment (SAGE)[3]. Basically, what SAGE does is merge many smaller displays into one large display and provide an architecture to control those displays as one. In the architecture a simple window management system is included opening every object in its own window. The display is controlled by an operator sitting behind a desktop system. Figure 1.2 shows the layout of the laboratory used at the University of Amsterdam.

The participating scientists need to ask the operator to display images or otherwise change the screen. Scientists other than the operator might not be able to do that, because they are unfamiliar with the interface for controlling the large display. A gesture-based interface could be of benefit to them in this environment giving them more freedom and flexibility, as studies in similar contexts show [4, 90]. The scientists could make gestures while standing in front of the

[3] See http://www.evl.uic.edu/cavern/sage/index.php. Last visited on January 14, 2010.

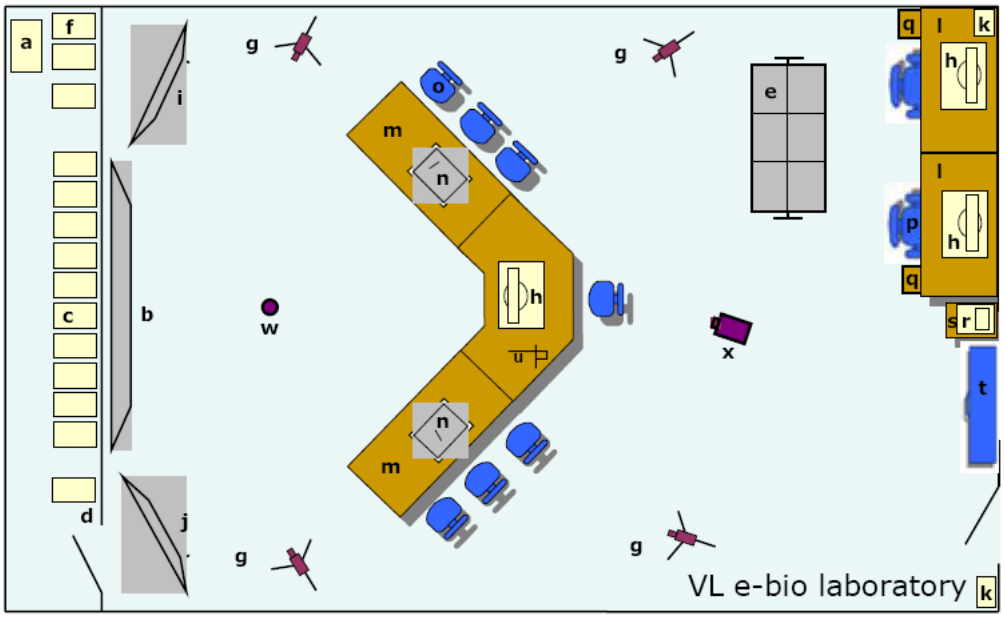


Figure 1.2.: *The layout of the e-BioLab, taken from [69].*

display to display the images they need or to manipulate them while discussing them with fellow scientists. This removes the need for an operator to control the display and the communication between scientists and operator, during which requests could possibly be misunderstood. It gives the scientists more freedom to control the display as they see fit, without having to trouble the operator. The ultimate goal would be to enable two or more scientists to control the display by making gestures simultaneously. The e-BioLab is taken as an example throughout this thesis to research the practical application of hand gesture interaction. This practical application is then extended to the use of controlling large display surfaces with hand gestures in general.

1.5. Method

The goal of this thesis is the user-independent recognition of dynamic gestures. Figure 1.3 depicts the method used.

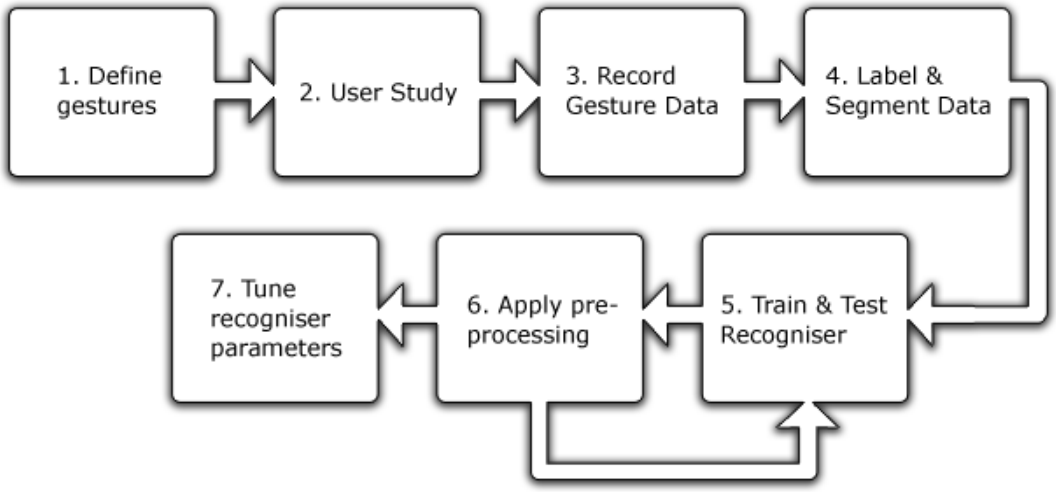


Figure 1.3.: *The method used (in phases). Phases five and six were repeated for each recogniser. Phases five, six and seven were repeated for each recogniser technology.*

In order to recognise dynamic gestures the first step was to define a dynamic gesture set based on the current literature and the context described in section 1.4. Having targeted a user group a small study was carried out to discover their characteristics. With the gesture set and results of the user study, a Wizard of Oz experiment was held [12], the goal of which was to create a data set containing instances of all gestures from as many users as possible. In this experiment the user seemingly controlled two applications by using the gestures from the gesture set, by which is meant that the user makes the gesture while it is being interpreted and translated into interface actions by the researcher. While the users were making the gestures the data they generated was recorded and globally segmented. The data was recorded using a dataglove that the users wore. This right-handed glove has several flexion sensors to measure how the wrist and fingers are bent. In order to incorporate the position and orientation of the hand in 3D space, a magnetic tracking sensor was attached to the glove. The data recorded through these devices formed the basis for the training of the recognisers. In order to make the data suitable for training, it had to be cleaned from noise and precisely segmented (i.e. the beginning, end and class of gesture had to be marked). Recognisers were trained and tested with the cleaned and segmented data. Seventy-five percent of the data was used for training. The remaining twenty-five percent was reserved for testing the percentage of successfully recognised gestures on data unseen during training. The effects on the recognition performance by tuning the recognisers through pre-processing of

the data set were studied in combination with variations in the parameter settings of the recognition technologies. Using this method a comparison was made between a traditional and a novel technology with regard to their recognition performance.

1.6. What this thesis is *not* about

This thesis does not address issues of intuitiveness and efficiency with regard to the use of hand gestures and how users experience them in this regard. Instead it focuses on the optimisation of the technical aspect of hand gesture recognition.

As several researchers have pointed out, it is important to define gestures which are intuitive, come natural to humans and are easy to remember [21, 86]. Although this thesis uses related literature to define a gesture set, it has a different focus and therefore does not contain research on the intuitiveness, naturalness and remembrance of individual gestures.

The gestures recognised are from a single hand only. Recognising gestures made by two hands would be ideal and would add to the gesture interaction possibilities [74]. Unfortunately, it would also complicate the research in such a way that it does not agree with the time frame for this thesis.

1.7. Thesis structure

This thesis is structured as follows. Chapter 2 continues with the definition of a dynamic gesture set suitable for controlling large display surfaces at a distance. Consecutively, Chapter 3 explains in detail the Wizard of Oz experiment setup and the properties of the recorded data. The technologies that were researched and applied to user-independent, dynamic gesture recognition are described in Chapter 4. In Chapter 5 the performance results obtained during the recogniser experiments are analysed and explained. The significance of these results is interpreted in Chapter 6. Lastly, recommendations for future research are made in Chapter 7.

Chapter 2

Tasks and Gestures

This chapter describes the dynamic hand gestures and how they were defined. The gestures were found by researching the tasks users do when controlling a large display surface. Section 2.1 describes the context of this research, finds a list of tasks specific to that context and shows that the tasks are context independent. In Section 2.2 a detailed analysis of the tasks addresses the question if these could be translated to gestures without any change and Section 2.3 concludes by describing the hand gestures as they are finally used.

2.1. Tasks in the e-BioLab

In Section 1.4 the e-BioLab was described in which a group of multidisciplinary scientists work together to address questions in the fields of life sciences. One of the tools available in the lab is a large display surface, which is used to visualise these datasets. Currently, the display is controlled by an operator from a separate desktop or small tablet PC. The operator is a scientist specially trained in operating the system and software used to control the display. If other scientists in the lab want to change what is being displayed they need to go via the operator. This is where a hand gesture interface could improve the method of control..

Driven by the possibilities and advantages of such an interface, this example and its context were further explored in this thesis. An initial list of user tasks was compiled from Kulyk et al. [41] and Rauwerda et al. [69], describing a formative user study, the e-BioLab and its setup. Only user tasks which involve controlling the large display were considered. In this context the term ‘user task’

is defined as described in Stone et al. [80, ch. 4, § 1.1]: “A task is a structured set of related activities that are undertaken in some sequence." The initial list of tasks, structured in two categories is shown in Table 2.1.

Image viewing	**Volume viewing (3D)**
• Panning	• Rotating
• Zooming	• Zooming
• Selecting	• Selecting

Table 2.1.: *Initial list of user tasks for controlling the large display in the e-BioLab.*

Revisiting this list in an interview with the operator of the e-BioLab resulted in the more complete listing as shown in Table 2.2.

Tasks currently possible	**Tasks possible in the future**
1. Window management	1. Volume viewing (3D)
a) Resizing	a) Rotating
b) Repositioning	b) Zooming
c) Minimising and maximising	c) Switching between 3D view modes
d) Closing	d) Switching between model layers
2. Image viewing	
a) Panning	
b) Zooming	

Table 2.2.: *Modified list of tasks after having discussed Table* **??** *with the display operator.*

This list is further divided into two new categories: tasks that are technologically possible and currently used in e-BioLab sessions and tasks that are currently not (yet) technologically possible, but are desired for the future. An additional difference with Table 2.1 is that the window management tasks were added. Amongst those are the tasks for resizing, repositioning and closing windows equal to that in traditional Windows-Icon-Menus-Pointing (WIMP) interfaces [20, 88].

This thesis limits its research to these subcategories, because they are actually used in the e-BioLab. Possible integration of the interface with the existing lab systems in the future is for that reason easier as well.

The applicability of the window management tasks are not limited to the e-

BioLab. They are easily imaginable in other contexts involving interfaces with window management systems. The same holds for the image viewing tasks. Therefore, we can define the tasks of Table 2.2 independent from any context as shown in Table 2.3.

Tasks currently possible

1. Window management
   - **Resizing** Enlarging and/or shrinking the size of a window on the large display, given that the window is not minimised.
   - **Repositioning** Moving the window left, right, up and down the 2-dimensional axes of the large display, given that the window is not minimised.
   - **Minimising** Hide the contents of the window and reduce it to the form of a small bar at the bottom of the display, given that the window is maximised or demaximised.
   - **Deminimising** Restore the window to its previous position and size and show its contents, given that the window is minimised.
   - **Maximising** Enlarge the window across the whole display, stretching its contents, given that the window is minimised or demaximised.
   - **Demaximising** Restore the window to its previous position and size, given that the window is maximised.
   - **Closing** Completely close the window, so that it is not in any way present on the display.
2. Image viewing
   - **Panning** The 2-dimensional repositioning of an image that is bigger than the window it is viewed in, so another, invisible part of that image is displayed.
   - **Zooming in** Enlarging an image from a specific point of focus in a window so as to see it from closer by and in more detail.
   - **Zooming out** Shrinking an image from a specific point of focus in a window so as to see more of the image and less detail.

Table 2.3.: *The tasks of Table 2.2, context independently defined.*

2.2. From e-BioLab tasks to elemental tasks

This section aims to translate the tasks from Table 2.3 into suitable hand gestures. Recall the definition of 'user task' from paragraph 2.1. The important part in this definition is that a task consists of a set of activities. This implies that a task can be further subdivided into activities, also called actions [80, ch. 4, § 1.1]. Applying this notion to the tasks already defined it becomes apparent that they

too consist of actions. The tasks window resizing and repositioning, for example have as precondition that a window needs to be selected before they can be executed. The other tasks in the list are similar. They all have certain preconditions which need to be satisfied before the actual task can be done. These preconditions form a part of the set of actions of a task. Until now the analysis and definition of these tasks were approached from a user perspective (hence, 'user task'). However to get tasks which can be directly translated into gestures it makes more sense to approach it from an interface perspective. Although that perspective uses a different terminology, the definitions remain the same. For example, the term 'user task' from the user perspective is termed 'general interface task' in the interface perspective. The term 'action' is termed 'elemental interface task', which are tasks defined at such a low level that they can be executed directly. They have no preconditions which need satisfaction. Fikkert [20, § 3.2] describes them as being at the heart of interfaces: "They build up an interface by being repeated throughout the various facets of the whole interaction." He also describes the best example being the Windows-Icon-Mouse-Pointer (WIMP) design. In this design the elemental interface tasks (also elemental tasks) of point-and-click events are used and reused over and over again.

From this it follows that elemental tasks can be translated into hand gestures. This means that the general tasks need to be subdivided one level further to identify the elemental tasks. Before that is done however, the problem of misunderstanding should be addressed. Misunderstanding about when the user makes a gesture which should be interpreted by the interface and when not. Buxton [5] describes three states in which the input can be. Alternating between the first two describes the problem of misunderstanding. Buxton calls the first state the zero or 'out-of-sync' state and the second state the first or 'Tracking' state in which no input is given and the system tracks a user's input. Now take the following gesturing example, a user zooms in on an image and afterwards points out a particular part to colleagues. The zooming in task should be interpreted, however pointing should not. The interface should change from the zero state to the tracking state while zooming and returning to the zero state afterwards. However, the interface should stay in the zero state while pointing out the parts to colleagues. There is no clear signal or cue given when to change from zero to tracking and back [5]. This problem is not unique to gesture interaction. Speech interaction has the same problem in the form of the interface not clearly knowing when its addressed and when not. There is no clear difference between when to interpret and when not, then a gesture or spoken utterance should be

interpreted in one context, but not in another. Keeping contextual information could play an important part in solving this problem. A solution to this problem for gesture interaction might also contain the use of various interaction zones a user stands in. Interaction zones define the role users play in a collaborative setting [24]. However, to research such a solution is outside of the scope of this thesis. In this research an on/off switch available on the hardware is used. Naturally, gestures are interpreted when the hardware is switched on and stops when it is switched off. This provides two extra elemental tasks, activate and deactivate.

Returning to breaking down the general tasks in elemental tasks, Table 2.4 gives an overview. In this overview it is assumed that only a single task is executed.

#	General interface task	Elemental interface tasks
1	Resizing	activate - point - select - enlarge / shrink - deactivate
2	Repositioning	activate - point - select - move - deactivate
3	Minimising	activate - point - select - minimise - deactivate
4	Deminimising	activate - point - select - deminimise - deactivate
5	Maximising	activate - point - select - maximise - deactivate
6	Demaximising	activate - point - select - demaximise - deactivate
7	Closing	activate - point - select - close - deactivate
8	Panning	activate - point - select - move - deselect - deactivate
9	Zooming in	activate - point - select - zoom in - deselect - deactivate
10	Zooming out	activate - point - select - zoom out - deselect - deactivate

Table 2.4.: *Overview of general interface tasks and their corresponding elemental interface tasks.*

The reader notices that tasks 1, 9 and 10 and tasks 2 and 7 have basically the same elemental tasks, see Table 2.4. Although they are used in a different context, the result is the same. The names of the general tasks appear to suggest a different meaning, however at the elemental level they are identical. Incorporating these facts in a new, merged overview, it is reduced to that in Table 2.5.

#	General interface task	Elemental interface tasks
1	Resizing / zooming in / zooming out	activate - point - select - enlarge / shrink - deactivate
2	Repositioning / panning	activate - point - select - move - deactivate
3	Minimising	activate - point - select - minimise - deactivate
4	Deminimising	activate - point - select - deminimise - deactivate
5	Maximising	activate - point - select - maximise - deactivate
6	Demaximising	activate - point - select - demaximise - deactivate
7	Closing	activate - point - select - close - deactivate

Table 2.5.: *Merged overview of general interface tasks and corresponding elemental interface tasks.*

Figure 2.1 shows a model in which the elemental tasks are generalised in how a single task is executed.

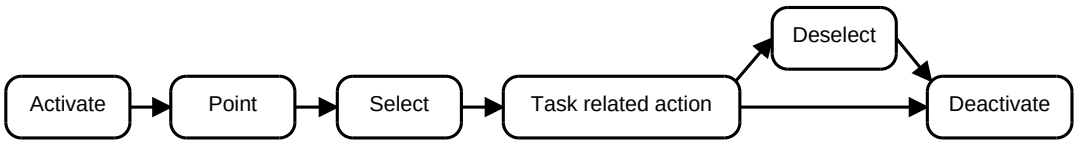


Figure 2.1.: *A model for executing a single general task.*

The model specifically shows the execution of a single task. Naturally, general tasks are executed consecutively by users. This makes the constant activation and deactivation in every task of the gesture interface superfluous. Therefore it is reduced to just one activation and deactivation at the beginning and end of the interaction.

As Figure 2.1 describes a model for a single general task, how would a model look like that incorporates all elemental tasks and the possible transitions between them? A model which in fact describes the possible inputs and their consequences. Such models exist for graphical input and several were described in existing literature. Buxton [5] describes such a model having three states in which the graphical input can be. This model can for example represent point, select and drag interactions for indirect input devices like mice, but direct input devices like touch screens also. However, this is different for free-hand direct interaction as Fikkert [20] describes. He denotes the three states from Buxton's model as: out of range, tracking and selected, which is equal to deactivated, pointing and selected in Figure 2.1. For the elemental tasks activate, point and select a separate state is created. The remaining elemental tasks described in

Table 2.5 are not modelled in Buxton's model. Those tasks can basically be considered as different manipulations in an interface, which is what Fikkert does. Furthermore Fikkert [20] describes that various extensions to Buxton's model were made in the past to model extra elemental tasks. However, most of these extensions fall short in their capacity to generalise over different interface manipulations and that is what is necessary with direct free-hand interface input. Fikkert solves this shortage by following the approach of Hinckley [28]. Hinckley's approach introduces a new state in the model for every new mouse button, in which every mouse button introduces a new form of manipulation. The disadvantage is that the model grows fast, because every new state introduces new state transitions with the existing states. To prevent this, Fikkert extends Buxton's model with a fourth state. In this state all manipulation tasks are modelled dynamically. The fourth state is implemented in a way such it adapts to the task. In this way all remaining elemental tasks besides activate, point and select can be modelled. Figure 2.2 shows the four-state model taken from Fikkert [20]. It also shows the 'Manipulating' state having various 'substates' representing the types of manipulations. The possible transitions between states are represented by the arrowhead-arcs in this thesis.

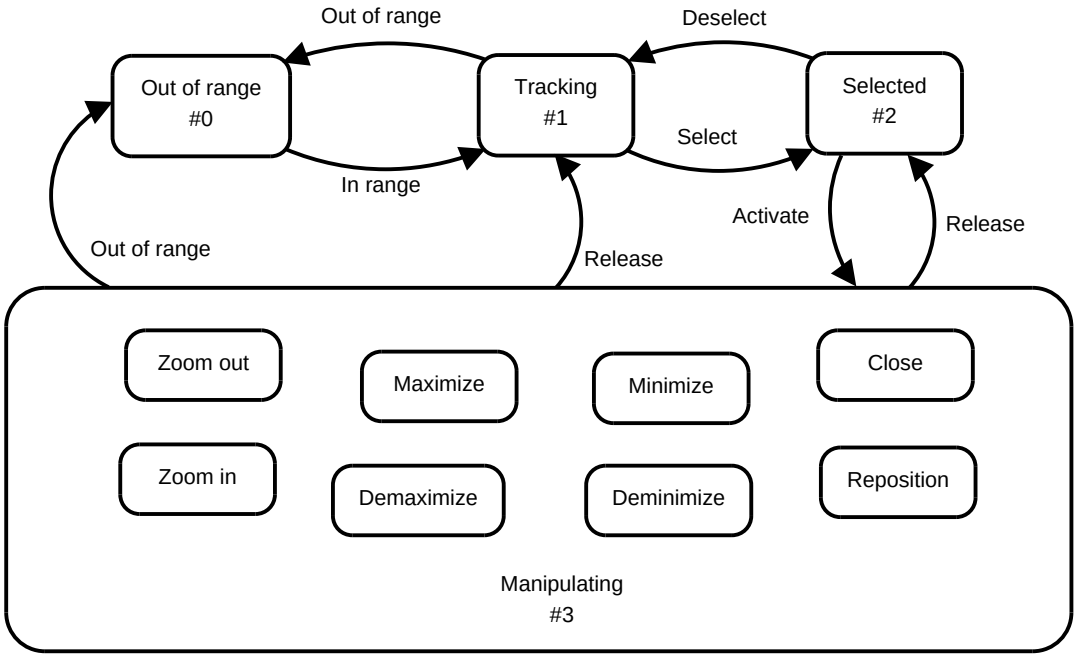


Figure 2.2.: *Four-state model taken from Fikkert [20] in which the types of interface manipulations from this research are represented as 'substates' of the fourth state. The substates are mutually fully connected, but to reduce clutter the arrowhead-arcs are left out.*

2.3. From elemental tasks to gestures

The goal of this chapter, to have a set of hand gestures defined for general tasks, is achieved in this last paragraph. Table 2.6 lists the twelve elemental tasks for which gestures need to be defined.

- Activate
- Deactivate
- Point
- Select
- Deselect
- Enlarge / Zoom in
- Shrink / Zoom out
- Maximise
- Demaximise
- Minimise
- Close
- Reposition

Table 2.6.: *Exhaustive list of elemental tasks to define gestures for.*

Multiple gestures for doing many of the elemental tasks above are defined by Fikkert [20][21]. In various experiments he evaluated the intuitiveness of gestures and which gesture users would use when confronted with in real life. The following subparagraphs describe per pair of tasks (e.g. activate / deactivate, minimise / deminimise etc.) the hand gestures chosen. The decisions are based on the results described in the previously mentioned literature.

2.3.1. Activate and deactivate

Several 'gestures' have been thought of for activating and deactivating the interpretation of gestures by the interface. The first option that came to mind was the user moving his hand above a certain threshold height regarding his body to activate the interface. However, this is physically hard to endure for longer periods. Additionally, one goes above the threshold pretty quick if the user just likes to point out something on the display without having it interpreted as a gesture. As described in section 2.2 making use of different interaction zones is another option. Although this option may have great potential it is beyond the scope of this thesis. For the remainder of this research a button is relied on to activate and deactivate the interface according to the user's needs. An on/off switch located on the hardware is used for this purpose.

2.3.2. Point

Pointing comes down to controlling a pointer in two dimensions on the large display surface. This has the similar concept of that of a mouse controlling a

pointer in the WIMP metaphor. The hardware used provides for a 3-dimensional position signal. This signal can be used to control a pointer on a 2-dimensional surface in real-time. However, the pointer should not be moved when a user makes another gesture. Therefore the pointer can only be controlled when in the pointing mode. This mode is active when the user shapes his hand in the form of a pistol in the upright position. In this way a user can immediately do a select to minimise the delay in the workflow, then pointing and selecting is often used in conjunction. Pointing itself is based on ray-casting as experiments from Fikkert [20] have shown, users rated this highest on both intuitiveness and 'would use'. If the user's hand is not shaped in a pistol the pointer remains on the last pointed location. This way the user can also point out objects on the display without having the interface interpret it as a gesture. Figure 2.3a visualises the pointing gesture.

2.3.3. Select and deselect

For selecting the results of the experiments in [20] have shown that the AirTap is rated the highest on intuitiveness, physical strain and 'would use'. The ThumbTrigger gesture does not fall behind much. Although the AirTap is rated highest, the ThumbTrigger better suits the context of this research, considering the hardware used. Based on this fact the ThumbTrigger gesture is used for selecting. The hand is shaped like a pistol and the thumb taps the middle finger to do a select task.

In regard to deselecting, the results show that the DropIt (stretched hand, palm facing downwards) and Select other (select background, empty space or other object) gestures are rated highest. Considering the fact that the focus of this research is on the technical realisation of gesture recognition, the choice is made to use the Select other gesture. It is equal to the gesture chosen for selecting, resulting in recognising two elemental tasks with one gesture. Additionally, the users mentioned that they were familiar with this concept of deselecting from the WIMP style interfaces. See Figure 2.3b for a visualisation of the gestures.

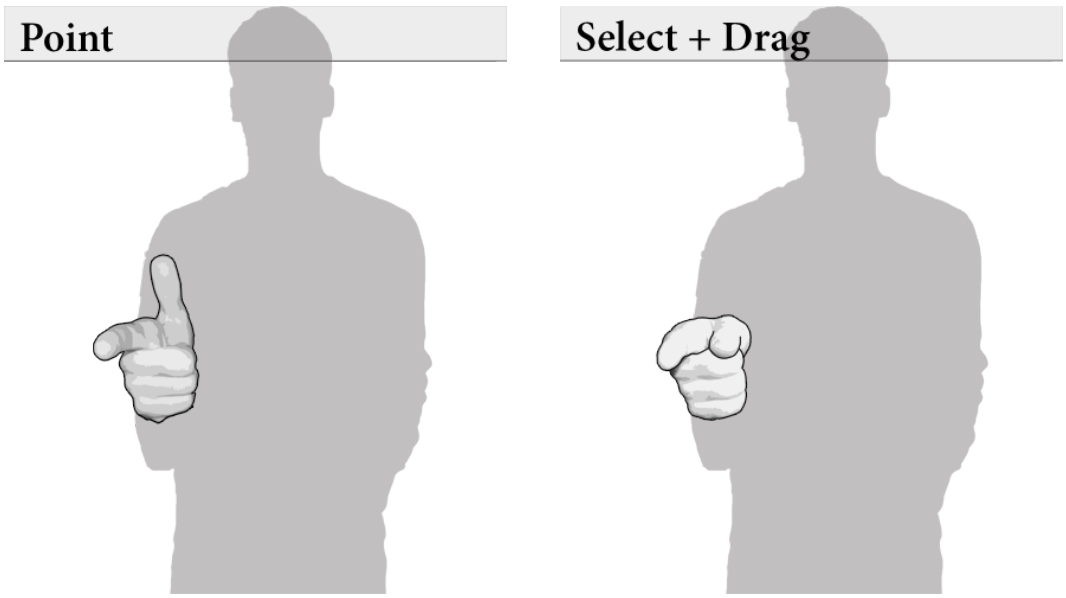


Figure 2.3.: *Gestures for Pointing (2.3a) and Selecting + Repositioning (e.g. Dragging, 2.3b).*

2.3.4. Reposition

Basically the gesture for repositioning objects on a large display surface is the same as that of pointing only with a little extra. The movement of the hand across the x and y axes of the large display enables the selected object to be repositioned. Note the words 'selected object' in the previous sentence. In fact this gesture uses position and select gestures to accomplish its goal.

2.3.5. Enlarge, zoom in and shrink, zoom out

For these elemental tasks Fikkert [20] also experimented with different gestures. Two gestures were found to be rated highest: 'Move hands apart' and 'Move fingers apart'. Given that this research is limited to gesturing with a single hand, the gesture 'Move hands apart' is removed as a candidate, leaving 'Move fingers apart' as the gesture of choice. This gesture can be done in multiple ways. By moving thumb and index finger from or to each other, by moving thumb and all other fingers from or to each other (as in grasping something between thumb and fingers). Additionally, the scaling factor of enlarging and shrinking (or zooming in or out for that matter) can be interpreted in two ways.

By repeatedly moving thumb and fingers from or to each other or by distributing the total scale across the whole distance between thumb and fingers. With the latter the user only has to move his thumb and fingers from or to each other once. Given that and that people are already familiar with the variant of this gesture for touch screens, the choice has been made to use thumb and index finger gesture with the scaling across the distance between them. Figure 2.4 visualises the gestures.

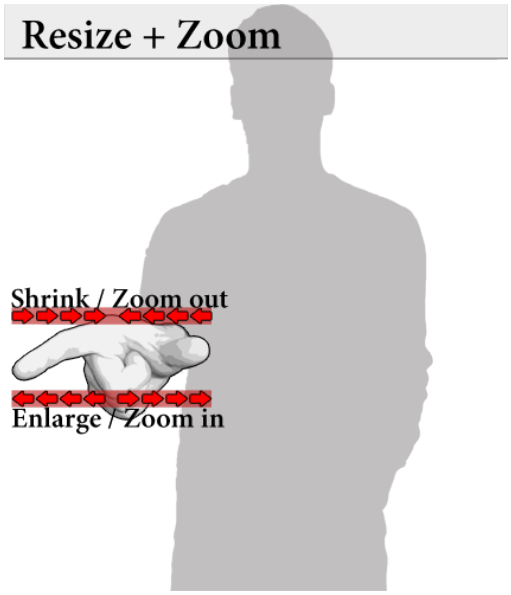


Figure 2.4.: *Gestures for enlarging / zooming in and shrinking / zooming out.*

2.3.6. Maximise and demaximise

Once again, this means pushing a window across the whole display (maximise) and to restore it from this state to its last size (demaximise). Fikkert [20] has not experimented with gestures for these elemental tasks. In existing literature nothing could be found on this subject as well. When thinking about possible one-handed gestures for these two elemental tasks, one tends to think in the direction of the enlarge and shrink gestures. Then to maximise and demaximise is basically 'enlarge to full screen' and 'shrink to previous size'. The difference is that no scaling distribution is needed. So, instead of moving thumb and index finger slowly from or to each other, moving thumb and other fingers explosively from or to each other might be intuitive and easy for the users to do. Moving thumb and other fingers apart explosively signals to maximise the selected window whereas moving thumb and other fingers to each other explosively until

they touch signals a demaximise. Partially these gestures are derived from enlarging and shrinking pictures in the interaction with multi-touch devices. See figures 2.5a and 2.5b for a visualisation.

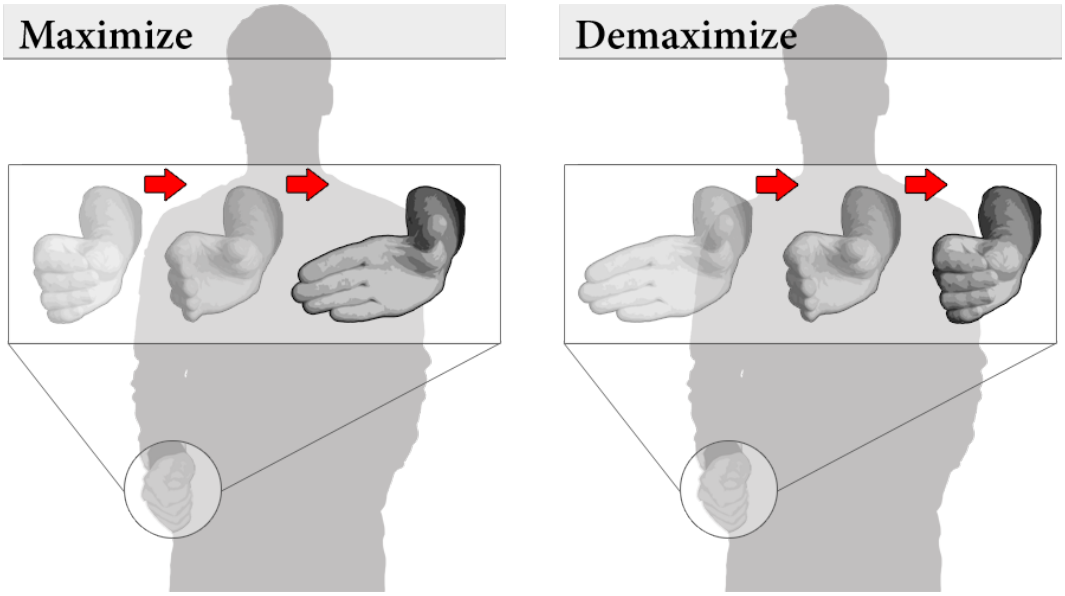


Figure 2.5.: *Gestures for Maximising (2.5a) and Demaximising (2.5b).*

2.3.7. Minimise and deminimise

Given that the goal of minimising is to send a window downwards hiding its contents only showing its titlebar, the gesture of waving it down seems intuitive and easy to use. The Apple iPhone also uses this gesture in a slightly different form for its touch screen. To deminimise the opposite gesture is used, wave the window up. These gestures also have an analogy in daily life in form of signalling an audience to sit down or stand up. Basically, one signals windows to 'sit down' or 'stand up' instead of an audience. A visualisation of the gestures is shown in Figures 2.6a and 2.6b.

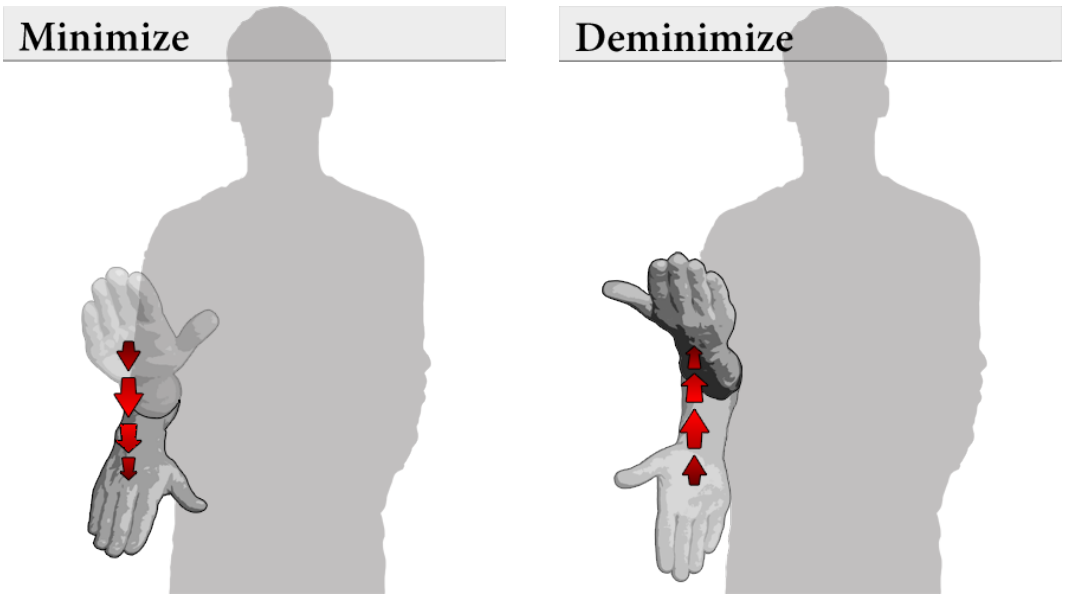


Figure 2.6.: *Gestures for Minimising (2.6a) and Deminimising (2.6b).*

2.3.8. Close

Closing windows by gesturing is also not much written about in current literature. When observing gestures people make in daily life, the closing of windows is comparable to something which is not needed anymore, can be removed or is disapproved to be shown on the display. To express this people in daily life often make a wavy gesture of disapproval or just waving it 'away'. With this gesture the hand is usually stretched in a relaxed way. Additionally, the thumb points upwards. Starting the gesture, the palm of the hand faces towards the user after which the wrist turns slightly to let the back of the hand face upwards in the end. See Figure 2.7 for a visualisation.

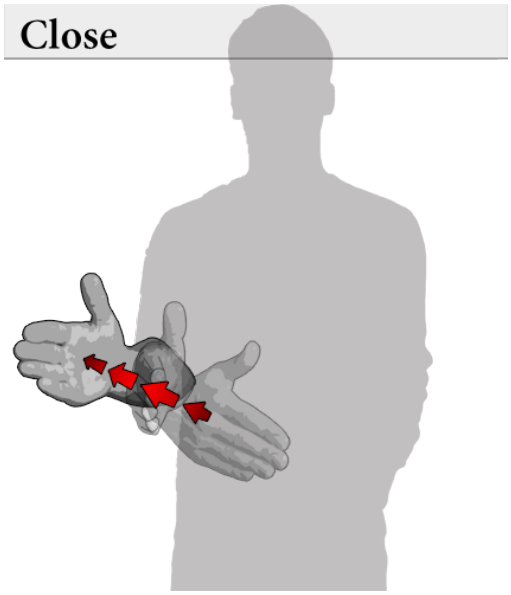


Figure 2.7.: *Gesture for closing.*

Chapter 3

Hand gesture data

Hand gesture recognition is a concrete case of a pattern recognition problem. To recognise patterns, samples of these patterns are needed to discover their distinctive characteristics. Without going into further detail here, patterns are recognised by 'matching' new patterns' characteristics to those from the trained samples. This means that samples of the gestures defined in Chapter 2 are needed to discover their characteristics.

This chapter is about how the necessary gesture samples were collected for training and testing a recogniser. A total of 2034 samples were collected for the nine gestures. Considering the purpose of this thesis is to recognise dynamic hand gestures user-independently, this thesis wanted to collect samples from as many users as possible. The following sections describe the sensors used for recording data, the experiment setup and the properties of the data set collected.

3.1. Used sensors

When considering the sensors used to recognise gestures the field of hand gesture recognition is generally divided into two categories [54]. Glove-based approaches use instrumented gloves to measure the flexion of the fingers, the position and orientation of the hand. The category of vision-based approaches uses cameras to record images of the user who is making gestures. In this category this thesis also includes marker-based solutions, since the markers are recorded through different types of cameras as well. Differences between these two approaches are described along several dimensions including accuracy, resolution, latency, range of motion, user comfort and cost. As Mitra et al. [54] and Mandel

[50] point out, the advantages of using a vision-based approach is that it does not hinder the ease and naturalness of the user's interaction. Nothing needs to be worn and no cables can hinder gesturing as compared to a glove-based approach where the user wears a glove and cables that attach the device to a computer. Advantages of glove-based approaches are that the hand and fingers are measured directly and the data can be provided at a higher sampling frequency. Vision-based approaches cannot track the hand in such detail without having to use computationally intensive calculations slowing down the tracking process. The first step in that process is always having to extract information from the images. Glove-based approaches on the contrary provide usable information immediately. Vision-based approaches also have to cope with line-of-sight occlusion, situations in which the hand is occluded by some object between the camera and the hand. Glove-based approaches do not have this disadvantage. Interference from the environment in which the tracking devices are placed, could also influence the recognition accuracy. The video stream from cameras could be distorted by high frequency signals or vibrating objects, for example. With most glove-based approaches additional sensors are used to track the 3D position and orientation of the hand. These are often based on ultra-sonic or magnetic technologies. Other ultra-sonic signals or metal surfaces from the environment could respectively interfere with the accuracy of these sensors.

Considering these advantages and disadvantages, a glove-based approach was chosen for this research. The main reasons for this are that datagloves provide direct measurement, no images need to be processed before the needed information is finally extracted. Additionally, datagloves provide the measurement of the hand posture without complex high-dimensional parameter reconstruction methods. From the dataglove used, signals of each sensor were read through a RS-232 interface device and put in an array of integers. As the research progressed, the discovery was made that further post-processing of the array of integers proved advantageous to the recognition rate. Refer to Section 5.2 for more details.

In this research a $CyberTouch^{TM}$ dataglove from CyberTouch Systems[1] was used. Figure 3.1 shows the version used in this thesis.

This glove has eighteen high-accuracy joint-angle measurement sensors. They measure the flexion and abduction of the fingers and the wrist and thumb rotation. Unfortunately, because of wear and tear, two sensors have broken down.

[1] See http://www.cyberglovesystems.com/hardware/products/cybertouch.php. Last visited on January 6, 2010.

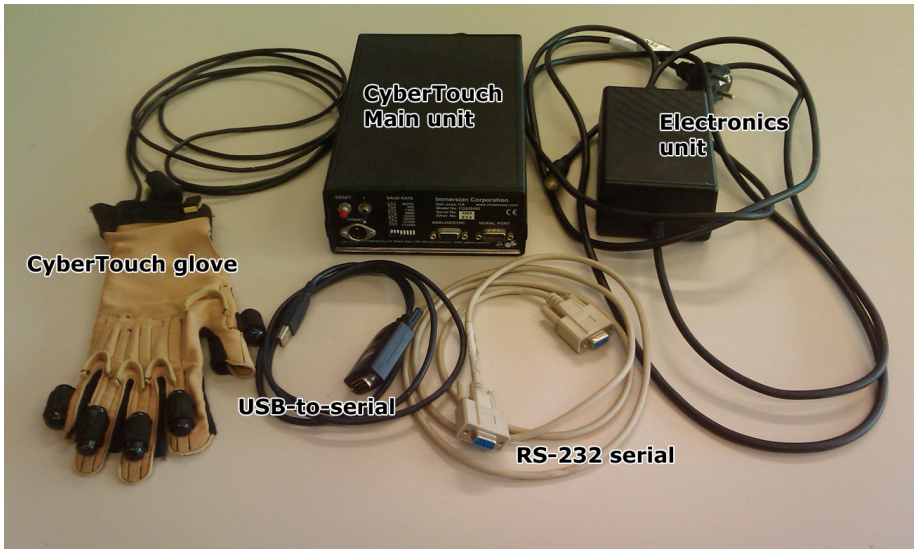


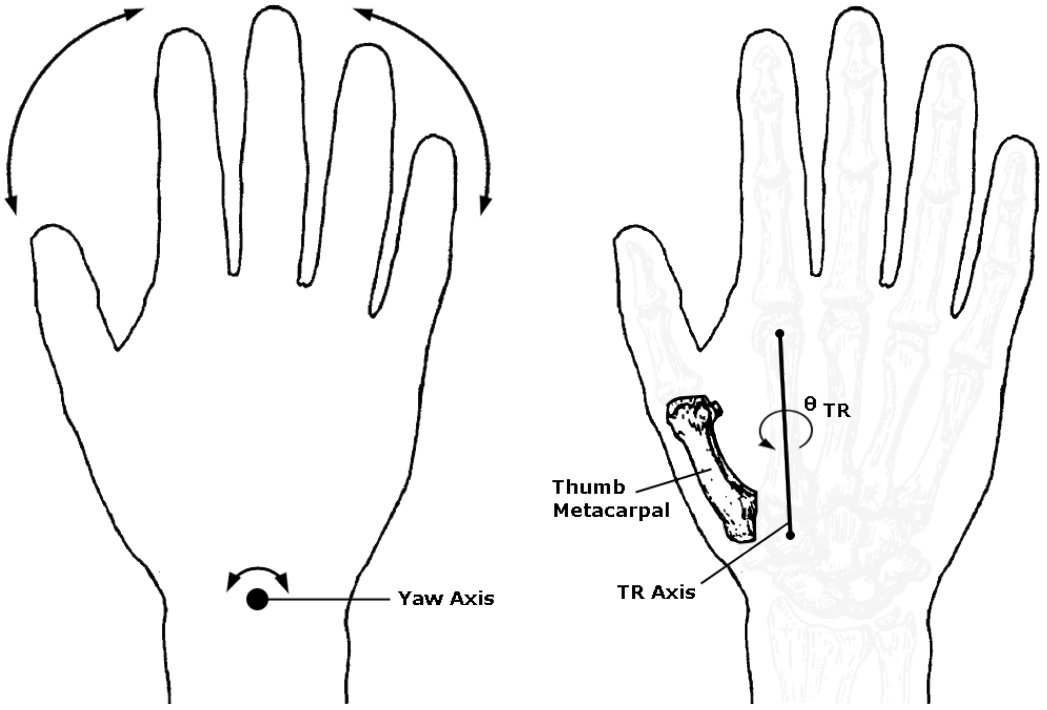
Figure 3.1.: *The $CyberTouch^{TM}$ dataglove, main unit and peripherals used.*

The wrist yaw and the thumb rotation cannot be used in this research. The wrist yaw measures the rotation of the wrist to the left or right with respect to the lower arm (see Figure 3.2a). Such rotation is found in a waving gesture for example. It is not found in any of the gestures in this research. The thumb rotation sensor measures the thumb rotation of the metacarpal bone, see Figure 3.2b for a description.

This rotation is found in the Select, Zoom in and Zoom out gestures when the thumb rotates from alongside to under the the index finger. Unfortunately, this could affect the recognition of the gestures, because without the thumb rotation they are more difficult to discriminate. These classes needed to be closely observed in the corresponding experiments. A total of sixteen sensors have been used. The glove also has 6 vibro-tactile actuators to provide feedback. However, these are not used in this research.

Considering the defined gestures, 3D position and orientation information is required of the lower arm. Unfortunately, the dataglove does not provide such information. Therefore the Flock Of Birds system[2] from the Ascension Technology Corporation was used. This system is able to measure the 3D position and orientation of an object by attaching sensors to them. It consists of a Standard Range Transmitter (SRT) unit that sends out a pulsed DC magnetic field, one ore more Fast Bird Bus units and one or more Flock Of Birds sensors [2]. See

[2]See http://www.ascension-tech.com/realtime/RTflockofBIRDS.php. Last visited on January 6, 2010.



(a) *The wrist rotates around the yaw axis, a rotation which is not used in any of the defined gestures.*

(b) *The thumb metacarpal (and the other thumb bones) rotates around the TR axis with value* ϑ_{TR}

Figure 3.2.: *Explanations of the wrist yaw and thumb rotations taken from [40].*

Figure 3.3 for the various components.

One sensor and a SRT connected to the FBB unit were used in this research. The dataglove has a special spot just below the wrist, where the Flock Of Birds sensor was attached. This way, the 3D position and orientation of the lower arm were measured. The actual output given by the Flock Of Birds system was configured to x, y, z position coordinates and x, y, z, w quaternion orientations.

3.2. Experiment setup

An experiment was designed to collect gesture samples of the nine classes of gestures. In this experiment users were asked to make gestures using the dataglove and Flock Of Birds sensor. To simulate a real hand gesture environment as much as possible, the experiment was held in one of the labs at the university that contained a large display surface. The display surface was generated by two projectors hanging from the ceiling next to each other. They projected on a wide,

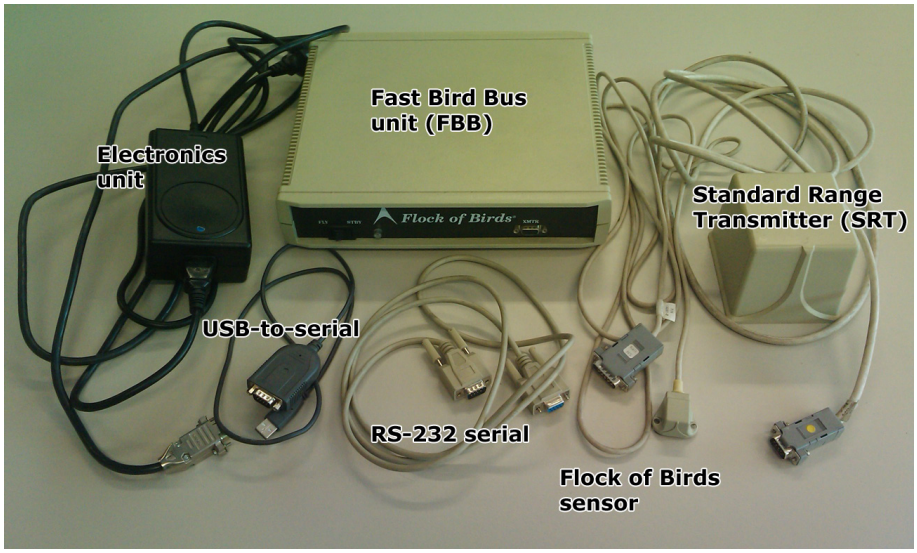


Figure 3.3.: *The Ascension Flock Of Birds system used.*

white screen that was made of special projecting cloth. The resolution used was 2560 pixels in width and 1024 pixels in height. The video signal came from two graphics cards in a quad core PC. Both the dataglove and Flock Of Birds were connected to this PC via USB-to-serial converters. The Flock Of Birds sensor was attached to the dataglove at the back of the wrist. Figure 3.4 shows the setup of the experiment room.

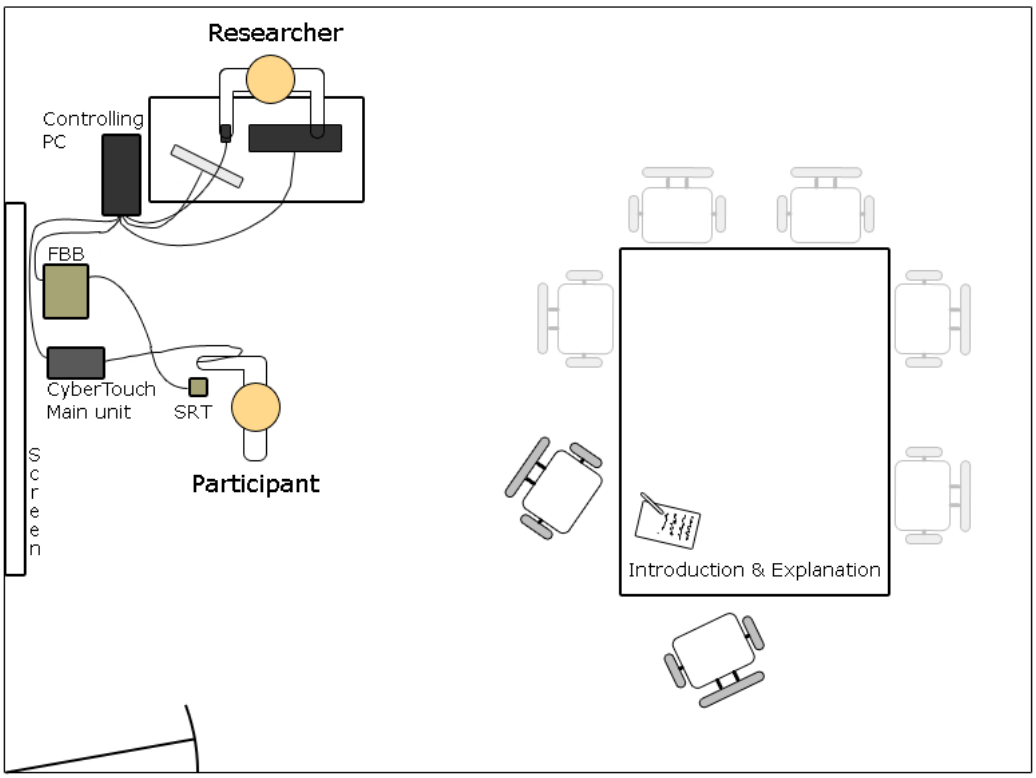


Figure 3.4.: *Layout of the recording experiment room. On the right the experiment was explained to participants. On the left, the gestures of the participant were recorded and interpreted by the researcher.*

The PC was running Windows XP and had 4 GB RAM. The data recording application was written in Java on top of the Squidy interaction library[3]. Squidy's goal is to provide a framework of extensions for integrating input devices, processing components (e.g. filters, 2D recognizers, etc.) and system actions, to ease research and development of novel interfaces. Squidy extensions were developed for the dataglove and Flock Of Birds. Additionally, a data recorder and log editor was developed to record and edit the gesture data. In a pipeline fashion, the dataglove and Flock Of Birds were connected to the log editor as input. The log editor was connected to the already existing Mouse I/O extension, so that the user could control the cursor on screen. In between a Kalman filter extension filtered the data on hand jitter. See Figure 3.5 for a screenshot of the Squidy setup of the above extensions.

The user controlled two applications with the defined gesture set. The first ap-

[3]See http://www.squidy-lib.de. Last visited on August 27, 2010.

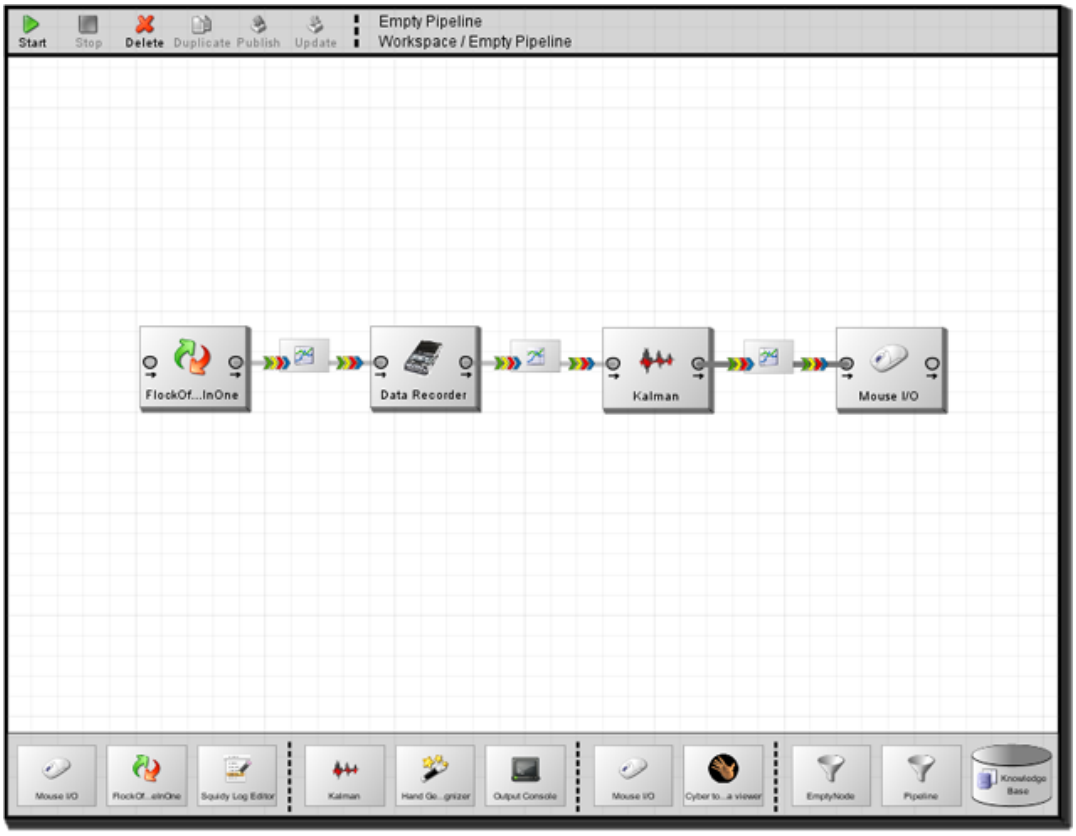


Figure 3.5.: *Screenshot of Squidy pipeline for recording gesture data.*

plication under control was the Firefox plugin CoolIris[4], a 3D photo wall with which photos can be selected, dragged, zoomed, brought to full screen, etc. A very interactive application suited for hand gesture interaction. With this application samples from the point, select+drag, maximise, demaximise, zoom in and zoom out gestures were collected. The second application was the card game memory, which was uniquely developed for this experiment in Adobe Flash. The goal was to match movie posters from a top and bottom row by selecting them, swiping them up or down in the 'memory machine' at the centre of the screen, wait for result and swipe the machine clean to start over. With this application samples of the point, select, minimise, deminimise and close gestures were collected.

The experiment was designed as a 'Wizard of Oz' experiment. In such an experiment a prototype system is developed, but without the actual automated recogniser module. That part is played by the 'Wizard', usually one of the re-

[4]See http://www.cooliris.com. Last visited on March 12, 2010.

searchers on the project. In general this type of experiment is commonly used to research how users react to the possibilities of new technologies before they are developed. In the field of gesture recognition it is common to collect data this way for training a recogniser. To collect that data a researcher needs to interpret the participant's gestures during recording.

During the actual experiment, when a participant entered the room, the researcher explained the goal of the experiment and the gesture set to use. The participant was assisted in putting on the glove, so it fits tightly. Afterwards, the Flock Of Birds sensor was attached to the dataglove at the back of the wrist. To align the sensor the participant was asked to stand near the magnetic field generator, so that the Flock Of Birds sensor was centred above it. This way, the centre of the magnetic field corresponded with the center of the screen. After having a short practice with the gestures on the CoolIris application, the recording started. Participants were encouraged to explore the applications on their own. Sometimes the researcher gave hints on what to do next if the participant did not know how to proceed.

The researcher sat behind the desk on which the computer was located and had a screen for himself to control the recording process. Additionally, the researcher pushed the correct keys on his keyboard after he saw a participant make a gesture as those described in section 2.3. That was done according to the protocol described in Appendix B. A crude segmentation and classification was also added to the recording log file while pressing the keys. When the recording was complete, the participant was assisted in taking of the dataglove and Flock Of Birds sensor. Afterwards the participant was thanked and received a candy bar of choice, as a small token of gratitude.

3.3. Data description

Data was recorded from a total of twenty-two participants. One of them is female. One is Iranian, two are German and nineteen are Dutch. Three participants are left-handed, which is notable because the dataglove is only made for the right hand. The quality of the data of these participants could be degraded because they had to do the gestures with their right hand. Furthermore, two participants are aged between 10-20, seventeen participants between 20-30 and three between 30-40. Considering their current occupation, one is a BA student, two are Postdoc researchers, three are Bachelor students at a University of Applied Sciences, four are Ph. D. students and twelve are MSc. students. Five

participants have no experience regarding hand gesture interfaces. One participant has experience with conventional mouse gesturing and one has experience with 3D mouse gesturing. Two participants have experience with mid-air bimanual laser-pointer gesture recognition and two have experience with mid-air hand gesturing through vision-based gesture recognition. Gesturing with the Wii gaming console[5] was experienced by Five participants. Fifteen have experience with either multi-touch phones or screens or both. For the full details of all participants see Appendix C.

From the twenty-two participants available, the data of six Dutch, right-handed, male participants was segmented and labeled in the scheduled time (participants no. 3, 4, 9, 15, 18 and 21 of Appendix C). Among them are one Ph. D. student, two Postdoc. researchers and three MSc. students. These were primarily selected because they gave the most usable and clear samples of every gesture which was determined after having reviewed a visualisation of the recorded gestures from all participants. This resulted in a total of 2034 gesture samples for the nine classes. Table 3.1 shows the number of gesture samples per gesture class.

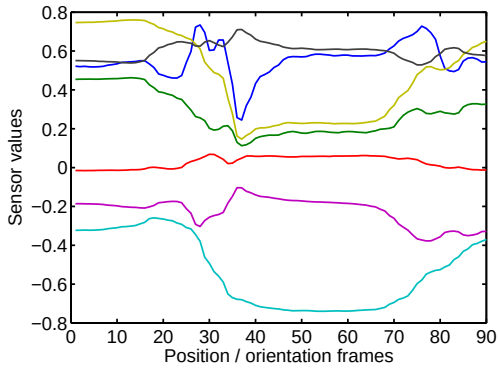
Class	**Close**	**Demax.**	**Demin.**	**Max.**	**Min.**	**Point**	**Select**	**Zoom in**	**Zoom out**
Samples	120	88	115	154	139	851	462	57	58

Table 3.1.: *Number of gesture samples per class.*

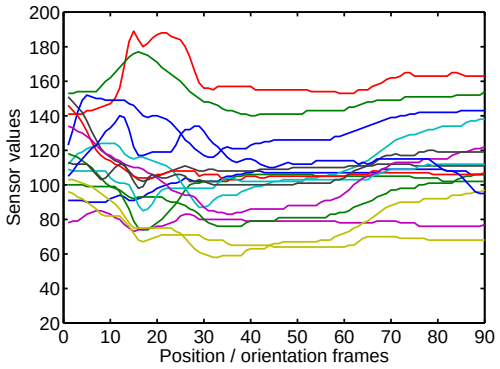
With gesture sample, a sequence of feature values is meant. They were recorded at around 70 Hz. One feature value has twenty-three dimensions. It contains three dimensions for the position feature, four dimensions for the orientation feature and sixteen dimensions for the hand posture feature. Both the position and orientation feature are in doubles and their range lies between $[-1, +1]$. The hand posture feature is in integers in the range of $[0, 255]$. Figure 3.6 shows a plot of two close gesture samples from the same participant.

Notice the differences between the two samples although they are made by the same participant. This could be an obstacle for a recogniser. Additionally, the data from the sensors are in two completely different intervals. This could also lead to poor recognition from a recogniser, because the recogniser might concentrate on the more prevalent data. To prevent this, it is better to have all features in the same interval. Another aspect of the data that is notable is the length of each sample. All samples have different lengths. That concerns samples within

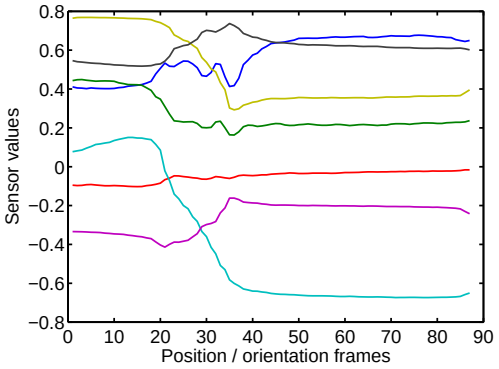
[5] See http://www.wii.com. Last visited on July 28, 2010.



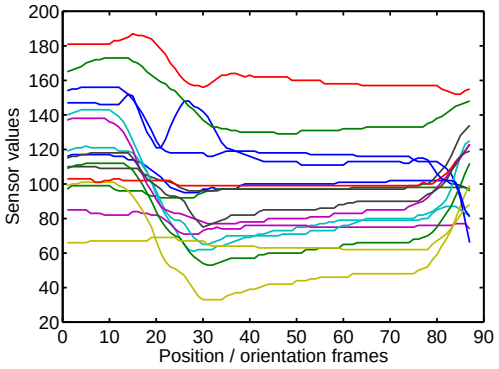
(a) *Position and orientation data of a Close gesture.*



(b) *Glove data of the Close gesture from 3.6a.*



(c) *Position and orientation data of a second Close gesture.*



(d) *Glove data corresponding to the Close gesture of 3.6c.*

Figure 3.6.: *Plots of two samples of the close gesture. Both are from the same participant. Figures 3.6a and 3.6c show the position and orientation data in their original* $[-1, 1]$ *interval. Figures 3.6b and 3.6d show the glove data in their original* $[0, 255]$ *interval.*

classes, but also between classes. Since the patterns are practically scaled variants of each other, it makes it harder for a recogniser to recognise.

Chapter 4

Recognition technologies

Considering gesture interaction from a machine perspective, machines and more specifically computers find themselves in a changing real-world environment. Changing in the sense that multiple users make gestures which the computer needs to recognise. These users all gesture in a different way, no matter how small that difference is. Humans seem to have the almost 'magical' ability to recognise the meaning of gestures. Computers (or machines in general) do not have this ability and need to learn the mapping between the input and the characteristics of gesture classes in order to recognise gestures. Amongst others the field of Human Computer Interaction uses machine learning for this purpose [1]. This way the computer that is in a changing environment is given the ability to learn. The advantage of this ability is that the interaction designer need not foresee and provide solutions for all possible situations [1].

In this thesis' case, the case of hand gesture recognition, the aim is to learn a mapping between a certain kind of input and output. In this thesis, the input is gesture data and the output the correct class of the gesture. Machine learning has different categories of learning applications. For example, Association learning [92, § 3.4], Classification [7], Regression [16], Supervised / Unsupervised learning [25] and Reinforcement learning [85]. Considering these applications the aim of hand gesture recognition in this thesis falls into the application of Supervised / Unsupervised learning. In supervised learning a mapping between input and output is learned from supervised data [1]. That data contains the correct values provided by a supervisor. The difference with unsupervised learning is that these correct values are not available, only the input data remains. Fortunately, that is not the case in this thesis. As described in section

3.3 the recorded input has been manually segmented and classified. With this data supervised learning techniques can be applied to learn the mapping between input data and gesture classes. Many different supervised learning techniques exist [6]. Considering the purpose of this thesis is to recognise dynamic gestures not all of these techniques are suitable. Because of the dynamic nature the techniques are required to handle data that changes over time. Additionally, a comparison was made between a traditional and more recent technique to see how recent developments in recognition technologies affect hand gesture recognition in this thesis' context. By these criteria Hidden Markov Models (HMMs) [13] were selected as the traditional technique and a more recent development is the technique of Latent-Dynamic Conditional Random Fields (LDCRFs) [57].

The above technologies are applied to develop and research a hand gesture recogniser. To develop such a recogniser the common process of supervised learning is applied. That process is briefly explained in this paragraph. After having created the data set described in section 3.3 the data set is divided into a training and test set. The gesture samples in the training set are used to train a recogniser according to the parameter settings specified. After the training is complete, the recogniser is tested on the gesture samples from the test set. This provides an objective test, since the recogniser has not seen the gestures in the test set. After all gestures from the test set have gone through the recogniser, the results in this research case are represented in the total recognition rate and a confusion matrix. The total recognition rate expresses the performance of the recogniser in the percentage of correctly recognised gestures. The confusion matrix displays how many gestures are classified in the available gesture classes. One row of the matrix specifies for that particular gesture how many instances of that gesture from the test set are classified (or misclassified for that matter) in the available classes. See Figure 4.1 for a visual representation of the recognition development process.

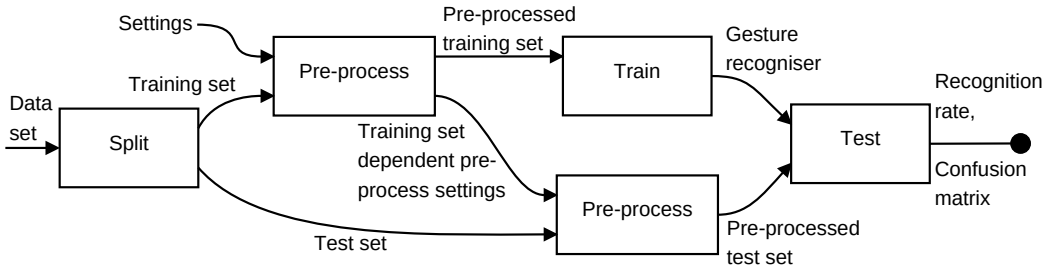


Figure 4.1.: *The recogniser development process with all inputs and outputs.*

The following sections in this chapter describe the principles of the applied recognition technologies, their characteristics and their parameters.

4.1. Hidden Markov Models

The theory behind HMMs was known for close to eighty years, but advances in the optimisation of Markov chains in the mid 1960s [3] were the cause of renewed interest in HMMs [67]. From the beginning of the 1970s HMMs are applied in different research fields ranging from speech recognition [22] to predicting financial variables [68] and from handwriting recognition [49] to nucleosome position prediction in bioinformatics [95]. HMMs have gained this popularity in all the different research fields because of their ability to take the temporal aspects into account of data that changes over time. In the course of the last fourty years many variants of HMMs have been researched [31, 34, 58].

A HMM is a statistical model. HMMs recognise patterns which are learned through observing sequences of a particular kind of data. It models a system which is considered to be in one of a set of a particular number of states. The system modelled is assumed to be a Markov process. A process is a Markov process when it qualifies the criteria that the conditional probability distribution of future states of the process, given the present state and a constant number of past states, depend only upon the present state. Additionally, in a HMM the states of the system are hidden, not observable. With not observable is meant that the state of the system is not known at any point in time [1, 67, 66]. Fortunately, observations, sequences of observations, O, are available that are generated by those hidden states. From these observation sequences the optimal hidden state sequence, Q, could be inferred given the Hidden Markov Model, λ. Let us now formally define the properties of a HMM. A HMM, λ, consists of a number of hidden states S, and the properties A, B and Π. $A = a_{ij}$, the state transition distribution between states S_i and S_j, $B = b_j(O_t)$, the observation probability distribution of emitting observation O_t at stateS_j and $\Pi = \pi_i$, the probability that state S_i is the initial state [66]. Given a number of observation sequences, the following three basic problems for HMMs are of interest [66]:

1. Given the observation sequence $O = O_1O_2...O_3$, and a model $\lambda = (A, B, \Pi)$, how is $P(O|\lambda)$, the probability of the observation sequence given the model efficiently computed?
2. Given the observation sequence $O = O_1O_2...O_3$, and the model λ, how is a corresponding state sequence $Q = q_1q_2...q_T$ chosen which is optimal in

some meaningful sense (i.e. best "explains" the observations)?

3. How are the model parameters, $\lambda = (A, B, \Pi)$ adjusted to maximise $P(O|\lambda)$?

This thesis is particularly interested in training a HMM from the observation sequences (i.e. gesture samples) recorded in the experiment of Chapter 3 (i.e. problem 3). In addition, recognising gestures from observation sequences given that model (i.e. problem 1) is also of interest.

Then, given an observation sequence, O, and a HMM $\lambda = (A, B, \Pi)$ the first of the basic HMM problems can now be solved. The observation sequence probability, $P(O|\lambda)$, is determined by means of the forward procedure. In this procedure, the forward α is calculated as follows [13, 66]:

$$\alpha_1(i) = \pi_i b_i(O_1), 1 \leq j \leq N \tag{4.1}$$

$$\alpha_{t+1}(i) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1}), 1 \leq t \leq T-1; 1 \leq j \leq N \tag{4.2}$$

In a similar way, a backward procedure exists in which the backward variable β is calculated [13, 66]. However, this procedure is only required in solving the third of the basic HMM problems. However, it is described here, because of its similarity to the forward procedure. The backward variable is calculated as follows:

$$\beta_T(i) = 1, 1 \leq j \leq N \tag{4.3}$$

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j), t = T-1, T-2, ..., 1; 1 \leq j \leq N \tag{4.4}$$

The calculation of the forward and backward variables often result in extremely small probabilities, going beyond the boundaries of machine calculation. Therefore, the forward and backward variables are scaled by a coefficient, c, keeping the calculation within machine calculation boundaries [13, 66]. The scaling coefficient is carefully determined in the following way:

$$c_t = \frac{1}{\sum_{i=1}^{N} \alpha_t(i)} \tag{4.5}$$

It is then simply applied to formulas 4.1, 4.2, 4.3 and 4.4.

The observation sequence probability, $P(O|\lambda)$, is finally determined by:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{4.6}$$

Before solving the third basic HMM problem a solution is needed to the second problem. To find the optimal state sequence associated with the given observation sequence David et al. [13] and Rabiner [66] et al. introduce a new variable:

$$\gamma_t(i) = P(q_t = S_i|O, \lambda) \tag{4.7}$$

The probability of being in state $S_i$at time t, given the observation sequence O, and the model λ. It can be expressed in terms of the forward-backward variables:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} \tag{4.8}$$

The third basic HMM problem is to determine a method to adjust the model parameters (A, B, Π) to maximise the probability of the observation sequence given the model. No analytical way exists to solve this problem. Therefore a solution is proposed that locally maximises $P(O|\lambda)$ by an iterative procedure as the Baum-Welch algorithm[13, 66]. This procedure introduces a new variable:

$$\xi_t(i, j) = P(q_t = S_i, q_{t+1} = S_i|O, \lambda) \tag{4.9}$$

The probability of being in state $S_j$at time t, and state $S_i$at time $t + 1$, given the model and the observation sequence. $\xi_t(i, j)$ can be written in terms of the forward-backward variables as follows:

$$\xi_t(i, j) = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)} \tag{4.10}$$

All required formulas are now defined to reestimate the A and Π HMM parameters [13, 66]:

$$\Pi = \pi_i = expected\, frequency\, in\, state\, S_i\, at\, time\, (t = 1) = \gamma_1(i) \tag{4.11}$$

$$A = a_{ij} = \frac{expected\, number\, of\, transitions\, from\, state\, S_i\, to\, state\, S_j}{expected\, number\, of\, transitions\, from\, state\, S_i} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{4.12}$$

Reestimating the B parameter depends on the observation density used. Subsections 4.1.1 and 4.1.2 define these for the discrete and continuous cases respectively.

To train a HMM, the number of hidden states needs to be defined. Hidden states try to capture and model the substructure of a gesture. With substructure, important and defining moments in a gesture is meant that largely discriminate a gesture. Hence, an important parameter, because varying the number of hidden states varies the substructure model. A second important parameter is the hidden state architecture. The number of hidden states can be arranged in different ways. They are connected like graphs in graph theory. Hidden states are the nodes in a graph that are connected to each other by lines representing the transitions. Two of the most common architectures used are the ergodic and left-to-right ones. In the ergodic architecture every state is directly connected to every other state, including itself (see Figure 4.2a).

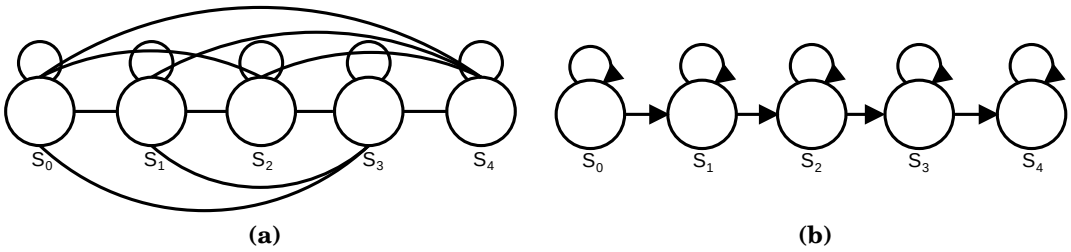


Figure 4.2.: *An example of the ergodic (4.2a) and left-to-right (4.2b) state architectures. Because of bidirectional transitions in Figure 4.2a the arrows are not shown.*

Left-to-right however applies an architecture in which the states are traversed in one direction (see Figure 4.2b). The underlying state sequence associated with the model has the property that as time increases either the state index increases or stays the same [66]. This is also called a BAKIS architecture. The effects the settings of these two parameters have on the recorded data set are studied in this thesis. The final goal is to find the optimal settings for this data set.

For every gesture class a HMM was trained. The nine HMMs were parallelly connected as proposed by David et al. [13]. Figure 4.3 shows an example.

The start and end states (i.e. S_{start} and S_{end}) describe the equal circumstances in which all the parallelly connected HMMs start and end. They are imaginary and serve no special purpose. Recognising a gesture involves calculating the

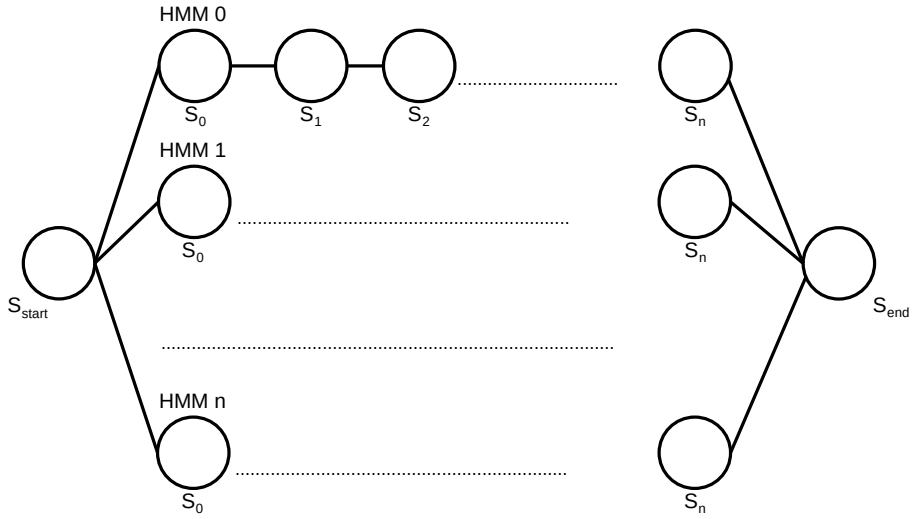


Figure 4.3.: *Example of the parallelly connected HMMs. The start and end states (i.e. $S_{start}$and S_{end}) connecting the HMMs are imaginary.*

observation sequence probability in every HMM. The gesture is classified in the class of the HMM that resulted in the highest probability.

In the comparison of recognition technologies two types of HMMs were included: Continuous HMMs (CHMMs) and Discrete HMMs (DHMMs). Both are trained and tested by algorithms which differ slightly. To train and test the HMMs the 'gpdsHMM' toolbox[1] for MATLAB[2] was used. This toolbox has algorithms to train both CHMMs and DHMMs.

The following subsections describe the formal theory behind DHMMs and CHMMs and the parameters their implementations provides.

4.1.1. Discrete Hidden Markov Models

Discrete HMMs differ from Continuous HMMs in that they handle discrete observations from a finite set of symbols. The continuous stream of data from the sensors is discretised in a finite set of symbols. The conventional way to discretise the continuous observation is by the process of Vector Quantization (VQ). VQ is a process in which all continuous observations are matched to a finite set of symbols, $\{v_k\}_{k=1,...,M}$ where N denotes the size of that set. Then the observation vector, O_t of the observation sequence, $O = O_1, O_2...O_T$ is obtained as follows

[1]See http://www.gpds.ulpgc.es/download/. Last visited on September 30, 2010.
[2]See http://www.mathworks.com. Last visited on September 30, 2010.

[13]:

$$\forall_{m \neq k} O_t = k \leftrightarrow d(x_t, v_k) < d(x_t, v_m) \tag{4.13}$$

with k being the index of the symbol v_k and $d(x_t, v_k)$ being the distance between x_t and v_k.

David [13] proposes the use of the Linde, Buzo and Gray algorithm (LBG) [47] in combination with the k-Means algorithm to calculate a finite set of N symbols from training data. A set of N symbols is calculated for each component in the observation vector. They try to find the symbol set size and vectors in which the overall distortion between the original observations and their quantisations is minimised. The proposed algorithm makes use of the 'Initial guess by splitting' [47] in which M-level quantisers are considered with $M = 2^R$ where $R = 0, 1, ...,$ and continues until an initial guess for an N-level quantiser with an acceptable level of distortion is achieved. The algorithm is as follows [87, § 2.2][47]:

1. Initialisation: Set $M = 1$ and calculate the centroid of this symbol for all observation vector components with all observation vectors from the training set.
2. Given the set of symbols containing M vectors $\{y_i;\ i = 1, ..., M\}$'split' each vector y_i into two close vectors $y_i + \epsilon$ where $\epsilon = 1 + randn(M) * DF$ and DF is the Distance Factor defining the maximum percentage of distance between the symbols of the current set with M symbols and the new set with $2M$ symbols.
3. If $M = N$, the initial guess for the N-level quantisation algorithm has been achieved. Else if $M < N$ run the k-Means algorithm for an M-level quantiser on the set of M symbols to minimise its Mean Square Error (MSE) and then return to step 2.
4. Run the k-Means algorithm for an N-level quantiser on the set of N symbols to minimise its MSE and then terminate.

k-Means is used to compute N centroids for the N symbols and iteratively updates the centroids given the training data. The k-Means algorithm is as follows [87, § 2.2]:

1. Initialisation: the N first training vectors are the $centroid(0)$.
2. Each added vector is assigned to the closest $centroid(t)$ and the $centroid(t+1)$ are recalculated with the new vectors added.

3. The algorithm is terminated when:

$$\sum ||centroid(t) - centroid(t+1)|| < threshold$$

$$or$$

$$iteration\,number > max\,number\,of\,iterations$$

The default values of $threshold$ and $max\,number\,of\,iterations$ are in the case of this thesis 0.005 and 10, respectively. These were taken from the toolbox [87] and not modified in this research.

With the above algorithms, hard matching decisions are made. Which means that the information about how observations match other symbols is discarded. Because of the large variability between the incoming observations those observations can be matched to such a displaced symbol in that this displacement is a source of misrecognition [13]. In an attempt to minimize that source of misrecognition David et al. [13] propose the use of 'Multi-labeling'. The difference with the conventional VQ method is that multi-labeling makes a soft decision about which symbol is closest to the incoming observation. Multi-labeling generates an output vector which components indicate the relative closeness of each symbol to the incoming observation. So, the multi-labeling finite set of symbols, N, maps the training vector x_t into an observable vector $O_t = \left\{w(x_t, v_{k)}\right\}_{k=1,\ldots,C}$, which is calculated with:

$$w(x_{t,}v_k) = \frac{1/d(x_t, v_k)}{\sum_{m=1}^{C} 1/d(x_{t,}v_m)} \tag{4.14}$$

Now that the discretisation process is defined, let us turn to solving the three basic HMM problems for DHMMs.

The first basic HMM, given an observation sequence, O, and a DHMM $\lambda = (A, B, \Pi)$, is solved with the previously described Equations 4.1 through 4.6. However, this research uses a discrete observation distribution with multi-labeling and therefore $b_j(O_t)$ is determined by [13]:

$$b_j(O_t) = \sum_{k=1}^{c} w(x_t, v_k) b_j(v_k) \tag{4.15}$$

In the implementation of the gpdsHMM toolbox, c, is defined as the L most significant values of $w(x_{t,}v_k)$ for each x_t where L is lower than C, from Equation 4.14. This reduces the computational load and makes the multi-labelling

approach to DHMMs more efficient [87, § 2.2.1]. The default value of c in the toolbox is 1. According to David [13], good results were obtained with the value of 1 for c (i.e. taking only the component of the closest symbol into account).

A solution to the second problem of finding the optimal state sequence associated with the given observation sequence is given by Equation 4.18.

The problem of (re)estimating the parameters of the DHMM, $\lambda = (A, B, \Pi)$, the third basic HMM problem, is solved by Equations 4.11 and 4.12 for Π and A after having calculated ξ by Equation 4.10. However, B is (re)estimated differently considering the discrete, multi-labeled observation distribution:

$$b_j(k) = \frac{\sum_{t=1}^{T} \gamma_t(j) w(x_t, v_k)}{\sum_{t=1}^{T} \gamma_t(j)} \tag{4.16}$$

With the above Equations DHMMs can now be trained and tested.

In previous research on gesture recognition with DHMMs the effects of the three parameters (i.e. number of states, architecture and number of symbols) have also been studied. Unfortunately, most of this research includes gesture recognition using computer vision and only limited includes recognition using datagloves. Although computer vision may employ a different kind of input method (usually one or two cameras), most of the features used are similar to those used from the dataglove. For this reason the results from previous research also including computer vision is used to guide initial parameter settings. In that research a four to ten state left-to-right architecture is mostly employed [97, 98, 72, 77, 78] However some approaches like Yamato et al. [96] utilize a larger number of hidden states (36). All approaches report above 90% recognition rate. As mentioned previously, little research is available on HMM glove-based recognition, however, Lee et al. [44] describe an approach using a five state left-to-right model. Properties of the gesture recognition process are often similar and whether to use an ergodic or left-to-right architecture and the number of states are often derived from an estimation of states which a gesture maximally could have [77, 78]. Some estimate the number of states by looking at the number of distinct features in the data [72]. Others just intuitively estimate a number of states and go from there [96, 97, 98]. All use these settings as initial values and try to improve on them empirically. In this thesis most gestures change over time. Considering that and assuming the fact that a gesture is somehow always completed before starting another, a left-to-right architecture is more appropriate, because of its constraint that it can only move through the

state sequence in one direction. When estimating the complexity of the gestures intuitively, the number of states the gestures need are estimated in the range of two to five states.

Next to these parameters the VQ process, unique to the DHMM, is guided by the parameter specifying the size of the finite set of symbols, N. The effects of this parameter were studied in this research. In previous research, Yamato et al. [96] recognised six human motion categories using seventy-two symbols and obtained 90.0% for user-dependent and 70.8% for user-independent recognition. Yang et al. [97] used 256 symbols in their recognition of digits drawn by following the path of the mouse. They obtained 99.78% recognition on a user-dependent isolated recognition task of nine gestures. Using fourty-eight symbols, Yoon et al. [98] obtained 93.25% with recognising hand gestures using their combination of weighted x-y-v features. Experiments with other features (e.g. location, angle and velocity) and combinations of these in different coordinate systems resulted in recognition from 41.79% to 92.96%. In those experiments the number of symbols ranged between two to fourty-eight. Considering these settings, the choice was made to use twenty symbols for the experiments in which the data pre-processing methods were tested. To tune the number of symbols in the final DHMM, the same strategy was adopted as Mäntylä [52] in which the number of symbols is doubled from sixteen until 256. Further tuning by increments of two was applied to the interval which showed the best results (see Subsection 5.3.1).

4.1.2. Continuous Hidden Markov Models

Continuous HMMs are characterised by the way they handle continuous observations. With Discrete HMMs, the symbols observed are quantified in a finite set. For Continuous HMMs, the distribution of the emitted symbols is continuous. It is assumed that the general observation can be represented in this research by a finite mixture of Gaussians, M, also the mixture component. Therefore, in the continuous case, $b(O_t)$ is defined as [13, 66]:

$$b(O_t) = \sum_{m=1}^{M} c_{jm}\mathcal{N}(O_t, \mu_{jm}, U_{jm}),\ 1 \leq j \leq N \tag{4.17}$$

Then, given an observation sequence, O, and a CHMM $\lambda = (A, B, \Pi)$ the first of the basic HMM problems is solved by Equations 4.1 through 4.5. The observation sequence probability, $P(O|\lambda)$, is determined by means of Equation 4.6.

The second basic HMM problem is solved by Equation 4.6, which for the continuous case is:

$$\gamma_t(i) = \left[\frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N}\alpha_t(i)\beta_t(i)}\right]\left[\frac{c_{ik}\mathcal{N}(O_t,\mu_{jm},U_{jm}}{\sum_{m=1}^{M}c_{ik}\mathcal{N}(O_t,\mu_{jm},U_{jm}}\right] \tag{4.18}$$

A solution to the third basic HMM problem, (re)estimating the model parameters of a CHMM, is given by Equations 4.11 and 4.12 for the parameters Π and A. B is (re)estimated in the continuous case according to the Equation 4.17. The following are the reestimation formulas for the parameters of the mixture of Gaussians (c and $\mathcal{N}(\mu, U)$) [13, 66]:

$$c_{jk} = \frac{\sum_{t=1}^{T}\gamma_t(j,k)}{\sum_{t=1}^{T}\sum_{m=1}^{M}\gamma_t(j,k)} \tag{4.19}$$

$$\mu_{jk} = \frac{\sum_{t=1}^{T}\gamma_t(j,k)O_t}{\sum_{t=1}^{T}\gamma_t(j,k)} \tag{4.20}$$

$$U_{jk} = \frac{\sum_{t=1}^{T}\gamma_t(j,k)(O_t-\mu_{jk})(O_t-\mu_{jk})}{\sum_{t=1}^{T}\gamma_t(j,k)} \tag{4.21}$$

Considering Equation 4.17 and the mixture of Gaussians reestimation Equations 4.19, 4.20 and 4.21 it can be seen that the Gaussian mixture component, M, influences the modeling of the general observation. This changes the HMMs ability to model the observation distribution, affecting its capability to recognise gestures. So, M is an additional parameter which needs to be tuned to optimise the recognition of gestures with CHMMs. Unfortunately, most of current literature on HMM hand gesture recognition uses DHMMs. For that reason no previous values for the mixture of Gaussians parameter unique to CHMMs were available other than one or two or 'more' [35, 34]. Additionally, similar to DHMMs, the number of states and state architecture parameters also apply to CHMMs.

To study the effects of these parameters on the recognition rate and to find the optimal settings, the method of testing as described in the introductory text of this chapter was used. These tests were run to test the combinations of parameter settings. Both the number of hidden states and the Gaussian mixture component were tested in the range of two to thirty with increments of two. Tests with combinations of these parameters were run with the ergodic and left-to-right state architecture.

4.1.3. Numerical instability

During the initial HMM tuning experiments, columns in the confusion matrices were observed which were completely zero. Numerical overflows appeared to be the cause which occurred during the training of the HMMs. The training algorithms used were the versions of the Baum-Welch algorithm for Discrete and Continuous HMMs. As described in Rabiner's tutorial on HMMs [66, Section V. - A.] both versions make use of scaling factors to avoid numerical underflow. The factors are the normalisation values of the forward probabilities for every timestep, t, per gesture instance. Both forward and backward probabilities are normalised with these factors, so that they cancel each other out when reestimating the state transition probabilities.

The case with the data set recorded in this thesis was that it contained gesture instances of more than 250 timesteps. Because of that length the multiplications for calculating the forward and backward probabilities also grew. With this growth the forward and backward probabilities become extremely small. As a consequence the scaling factors become extremely large, until they go beyond the upper boundary of machine calculation. That is where the overflow errors occurred and the training process derailed.

In addition, there was dissatisfaction with the recognition obtained with the training runs free from overflows, because recognition rates with HMMs in other contexts were consistently higher. Also, it was acknowledged that it could be because of multiple causes (too little training data, use of other HMM variants, too long observation sequences, etc.). However, the loss in precision was mainly believed to be the biggest influence. Therefore the numerical stable HMM training method described by Mann [51] was implemented. Mann gives a detailed description of an approach to deal with extremely small conditional probabilities. This approach entails working with the logarithms of those probabilities. To work with logarithms, four Equations are defined (see Equations 1-10 in Mann [51]). These Equations are essentially standard logarithm operations extended to correctly handle zero values. They must be handled, because events can have zero probability and taking the logarithm of zero is not a number. With these four basic Equations defined, the calculation of the forward (α), backward (β), gamma (γ) and epsilon (ϵ) probabilities were reimplemented according to algorithms five to nine in Mann [51] respectively. The reestimation of the initial state (π), state transition (A) and observation symbol probability distributions(B) were reimplemented according to algorithms nine to eleven.

4.2. Latent-Dynamic Conditional Random Fields

In many classification problems involving sequential data a solution is researched to predict a label for each frame. Additionally, these sequences of data are unsegmented having no marked beginning or end. Facing these problems in the area of computer vision, Morency et al. [57] proposed a new visual gesture recognition algorithm based on Conditional Random Fields (CRFs) [43]. Their Latent-Dynamic Conditional Random Fields (LDCRFs) aims to provide a solution for the problem of recognising individual classes of gestures and detecting the beginning and end of such gestures in unsegmented sequences. The HMMs of Section 4.1 only aim to recognise individual classes of gestures. Without extensions, the traditional HMM does not have the ability to detect the beginning and end of gestures in unsegmented sequences. This is particularly useful in real-time recognition of hand gestures and also provides the ability to further improve LDCRFs with unsegmented data once trained. As LDCRFs are based on CRFs, they too are discriminative models. Discriminative models provide a probabilistic model only for the target variables conditioned on the observed variables. In this research that is a mapping between a sequence of observations and a sequence of hand gesture labels. HMMs are generative models. These models provide a full probabilistic model for a joint probability distribution over observation and label sequences. However, to optimally learn such a model, the observations are assumed to be conditionally independent from each other. The main advantage of discriminative models is that this assumption is relaxed. An assumption that is otherwise too restrictive for data distributions of many classes.

LDCRFs can capture both subgesture patterns and dynamics between gesture classes, this is also termed intrinsic and extrinsic class dynamics respectively. “LDCRFs discover latent (i.e. hidden) structure that best differentiate visual gestures and distinguishes subtle motion patterns such as natural head nods and eye gaze aversion” [55, 57]. In comparison with previous discriminative models like Conditional Random Fields (CRFs) [43], LDCRFs incorporate hidden state variables modeling the substructure of gestures where CRFs do not. CRFs only model the transitions between gestures, thereby exclusively capturing the extrinsic dynamic. Hidden-state Conditional Random Fields (HCRFs) [65] are an extension to CRFs in which the use of hidden states was first introduced. Hence, LDCRFs combine the strengths of CRF and HCRF in capturing both intrinsic substructure and extrinsic dynamics.

Equal to the task of Morency et al. [57], this thesis also wants to learn a mapping between a sequence of observations and a sequence of labels. They give the following definitions [57]. Let a sequence of observations be $x = \{x_1, x_2, ..., x_m\}$ and a sequence of labels $y = \{y_1 y_2, ..., y_m\}$. Each y_j is a class label for the j^{th} observation frame of a hand gesture sequence and is a member of a set $\mathcal{Y}$ of possible class labels. Each observation frame is represented by a feature vector $\phi(x_j) \in R^d$, in this thesis' case, the 23 dimensional vector of 3D position, orientation and hand posture data. For each sequence a vector $h = \{h_1 h_2, ..., h_m\}$ of "sub-structure" variables is also assumed. Not observed in the training examples, they are the hidden variables in the model. Given these definitions, a LDCRF is defined (Equation 2 in [57]):

$$P(y|x, \theta) = \sum_{h:\forall h_j \in \mathcal{H}_{y_j}} P(h|x, \theta) \tag{4.22}$$

where $\mathcal{H}_{y_j}$ is a set of possible hidden states, h_j, for the class label y_j and $\mathcal{H}$, the union of all $\mathcal{H}_y$ sets. $P(h|x, \theta)$ is defined (Equation 3 in [57]):

$$P(h|x, \theta) = \frac{1}{\mathcal{Z}(x, \theta)} exp\left(\sum_k \theta k \cdot F_k(h, x)\right) \tag{4.23}$$

and the partition function, $\mathcal{Z}(x, \theta)$, as:

$$\mathcal{Z}(x, \theta) = \sum_h exp\left(\sum_k \theta k \cdot F_k(h, x)\right) \tag{4.24}$$

and F_k, the feature function as:

$$F_k(h, x) = \sum_{j=1}^{m} f_k(h_{j-1}, h_j, x, j) \tag{4.25}$$

Each feature function is either a state function $s_k(h_j, x, j)$ or a transition function $t_k(h_{j-1}, h_j, x, j)$, where state functions, s_k, depend on a single hidden variable, while transition functions, t_k, depend on pairs of hidden variables [57].

To learn the model parameters, Morency et al. [57] follow the initial defining work of CRFs [43]. The following objective function is used to learn the parameter θ^* (Equation 4 in [57]):

$$L(\theta) = \sum_{i=1}^{n} logP(y_i|x_i, \theta) - \frac{1}{2\sigma^2}||\theta||^2 \tag{4.26}$$

where the first term is the conditional log-likelihood of the training data and the second term the log of a Gaussian prior with variance σ^2. Gradient ascent with the Broyden–Fletcher–Goldfarb–Shanno (BFGS) optimisation technique is proposed to search for the optimal parameter values under the criterion, $\theta^* = arg\,max_\theta L(\theta)$. The gradients of both objective functions with respect to the parameters, θ_k, associated to either state function, s_k (see Equations five and six in Morency et al. [57]), or transition function, t_k, is shown to be computable in $O(m)$ using belief propagation [57, 61]. The transition functions are defined one for each hidden state pair $(h^{'}, h^{''})$ and are expressed as:

$$t_k(h_{j-1}, h_j, x, j) = \begin{cases} 1\, if\, h_{j-1} = h^{'}\, and\, h_j = h^{''} \\ 0\, otherwise \end{cases}$$

Weights, θ_k, associated with a transition for hidden states that are in the same subset, $\mathcal{H}_{y_i}$, model the intrinsic dynamics (i.e. substructure patterns). The weights associated with transition functions for hidden states from different subsets model the extrinsic dynamics (i.e. external dynamics between gesture classes) [57]. The number of state functions, s_k, are equal to the length of the feature vector, $\phi(x_j)$, times the number of possible hidden states, $|\mathcal{H}|$. In this thesis, a 23 dimensional feature vector is used with up to five number of states. The total number of state functions in the model used in this research is $23 \times 5 = 115$. Given a new test sequence, x, the most probable label sequence, y^*, is to be estimated that maximises the conditional model:

$$y^* = arg\,max \sum_{h:\forall h_i \in \mathcal{H}_{y_i}} P(h|x, \theta^*) \tag{4.27}$$

where the parameter values, θ^*, are learned from training sequences. The following algorithm is used [57]:

1. For all observation frames, j, from the observation sequence, y_j^*
   a) Compute the marginal probabilities $P(h_j = a|x, \theta^*)$ for all possible hidden states $a \in \mathcal{H}$ using belief propagation.
   b) Sum the marginal probabilities according to the disjoint sets of hidden states $\mathcal{H}_{y_j}$ into the log-likelihood for every gesture class.
2. Sum the log-likelihoods of all observation frames per gesture class.
3. Determine the maximum log-likelihood summed in step 2 and assign the associated label.

Just as with HMMs, computing the marginal probabilities (step 1a) for long observation sequences consists of many probability multiplications. That may cause numerical instability. However, the implementation used in this research solved this by replacing the multiplications by additions of the logarithms of the probabilities.

Since the introduction of LDCRFs in 2007 in the field of head gesture recognition they have also been applied in the fields of human action recognition [46] and speech recognition [81]. The inference algorithms of LDCRFs have also been further developed [82]. Besides these fields it seems that LDCRFs have not yet been applied in this thesis' research context. Since LDCRFs have proven to outperform traditional models in other, similar fields, results are promising for this thesis' research.

LDCRFs provide several parameters to tune it to the sequences modelled, effecting the final recognition performance. Similar to the CRFs, LDCRFs have a regularisation parameter. Regularisation is used to remove overfitting of the data by using a penalty on weight vectors whose influence in the total training process is too large [83]. Values for this parameter setting used in previous research [55, 57] have been 0.001, 0.01, 1, 10, 100, 1000, 10000 and 0. Like HCRFs, LDCRFs use a number of states to model the internal substructure of a gesture. Values used in previous research [55, 57] range from two to six hidden states. The third parameter is the window size. With this parameter the concatenation of feature vectors of multiple frames can be used to create the input feature used during training. For example, if the window size is one, only the feature vector of the current frame is used. If however, the window size is three, the input feature used is a concatenation of the feature vectors from three frames. Namely, the current frame, the one preceding it and the future frame. Values of the window size ranged from 1 to 31 in previous experiments [55, 57]. The effects of these parameters on the final recognition performance were studied in this thesis using the described values of these three parameters as guidance. In those studies the HCRF library v1.3d[3] of Morency et al. [57] was used.

[3] See http://sourceforge.net/projects/hcrf/. Last visited on October 5, 2010.

Chapter 5

Experiments and results

This chapter describes the results of the experiments carried out with the aim to increase the recognition rate (i.e. the percentage of gestures recognised correctly). Section 5.1 describes the results obtained with unprocessed data. The results obtained upon applying the pre-processing methods are described in Section 5.2. First the methods were applied one-by-one (Subsections 5.2.1 to 5.2.5) and subsequently various method combinations were investigated (Subsection 5.2.6). The effects of tuning the recogniser parameters were also studied (Section 5.3). In the final section of this chapter the results of our tests on user-independent recognition are described.

5.1. Unprocessed data

To obtain a baseline, experiments were first performed on the unprocessed data set. The following parameter settings were initially used to train the recognisers. For Continuous HMM (CHMM): four states, four Gaussians and left-to-right architecture; for Discrete HMM (DHMM): four states, twenty symbols and left-to-right architecture; and for the Latent-Dynamic Conditional Random Field (LDCRF): one hidden state, a regularisation factor of 10 and a window size of 0. All recognisers were trained using 75% of the data set. The remaining 25% was used as a test set. Each time a training was started the training and test sets were randomly chosen from the total data set. Three training / test runs were done for each setting. Table 5.1 shows the average recognition rates for the three recognition technologies.

Technology	Recognition rate
CHMM	35.31%
DHMM	43.40%
LDCRF	92.56%

Table 5.1.: *Recognition rates based on default parameter settings using unprocessed data.*

As the table shows, DHMMs performed a little better than CHMMs. However, both performed poorly compared to LDCRF.

5.2. Pre-processed data

Shalabi et al. [73] have researched the effects of pre-processing the training data to better support the training process of the recogniser. Their research shows that data pre-processing methods can positively affect the recognition rate. Chaturvedi et al. [8, § 4.2] have studied the performance of Artificial Neural Networks and found that the training performance depends on how the data is represented. Different methods of pre-processing are described by Priddy et al. [64, ch. 3] to obtain better results with automated recognition systems. For the purposes of this research the following pre-processing methods were used: rescaling [33], normalisation [64, ch. 3], interpolation [33], dimension reduction [33] and feature extraction [31]. The same parameter settings as described in Section 5.1 were used in training the recognisers. The results are shown for each method in the following subsections. The final subsection (5.2.6) describes the results obtained by combining pre-processing methods.

5.2.1. Rescale

As the values of the 3D position and orientation data are already in the interval $[-1, 1]$ the glove data is also rescaled to this interval. In the researcher's opinion, as the recogniser is trained on data in one interval recognising gestures should become easier. This way, the chance that recognisers focus themselves on data in a particular interval is eliminated. Both the training and test sets were rescaled. Table 5.2 shows the recognition rates using rescaled data.

Compared to unprocessed data, CHMMs performed considerably better. DHMMs and LDCRFs performed slightly better, but not significantly.

Technology	Recognition rate using unprocessed data	Recognition rate using rescaled data
CHMM	35.31%	43.89%
DHMM	43.40%	44.31%
LDCRF	92.56%	93.35%

Table 5.2.: *Recognition rates based on default parameter settings using rescaled data. The rates of Table 5.1 are shown for comparison.*

5.2.2. Normalisation

Normalisation smooths the differences between instances of the same gesture [9]. This makes the data independent of the sensor range. Although there are many data normalisation methods, this research is limited to min/max normalisation as it is the most commonly used method [33, 73]. With min/max normalisation, the minimum and maximum value of a feature column are determined and taken as -1 and 1 respectively. Subsequently all data in the column is normalised between these values [8, 64]. Equation 5.1 shows the min/max normalisation equation as proposed by Priddy et al. [64]:

$$y_i = (max_{\text{target}} - min_{\text{target}}) \times \left[\frac{(x_{\text{i}} - min_{\text{value}})}{(max_{\text{value}} - min_{\text{value}})} \right] + min_{\text{target}} \tag{5.1}$$

where y_i is the normalised value, min_{target} and max_{target} are -1 and 1 respectively. The values min_{value} and max_{value} are the respective minimum and maximum values of a feature column. Table 5.3 shows the resulting recognition rates.

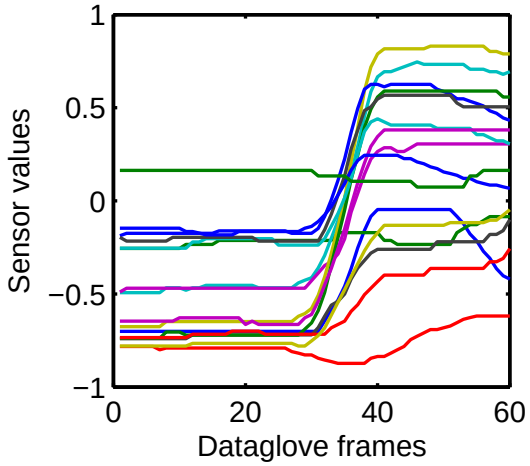
Technology	Recognition rate using unprocessed data	Recognition rate using normalised data
CHMM	35.31%	42.47%
DHMM	43.40%	46.02%
LDCRF	92.56%	91.39%

Table 5.3.: *Recognition rates based on default parameter settings using normalised data. The rates of Table 5.1 are added for comparison.*

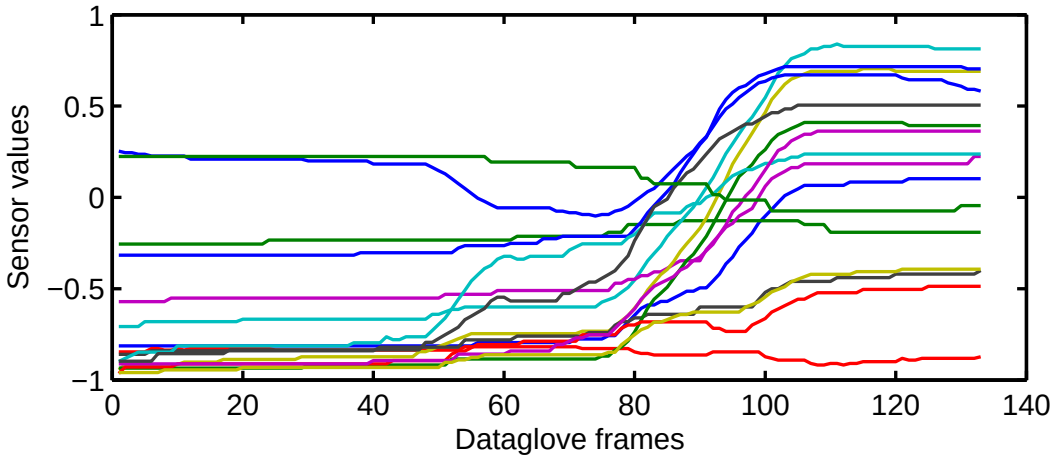
Compared to unprocessed data, the recognition rate of CHMMs improved significantly. DHMMs also improved slightly. LDCRFs performed slightly less well.

5.2.3. Interpolation

Gesture instances from the data set are of different lengths (i.e. the number of samples). For example, instances of the same gesture have similar patterns, but are 'stretched-out versions' of each other, as Figure 5.1 shows. This is a possible source of misrecognition for a recogniser.



(a) *An instance of the demaximise gesture.*

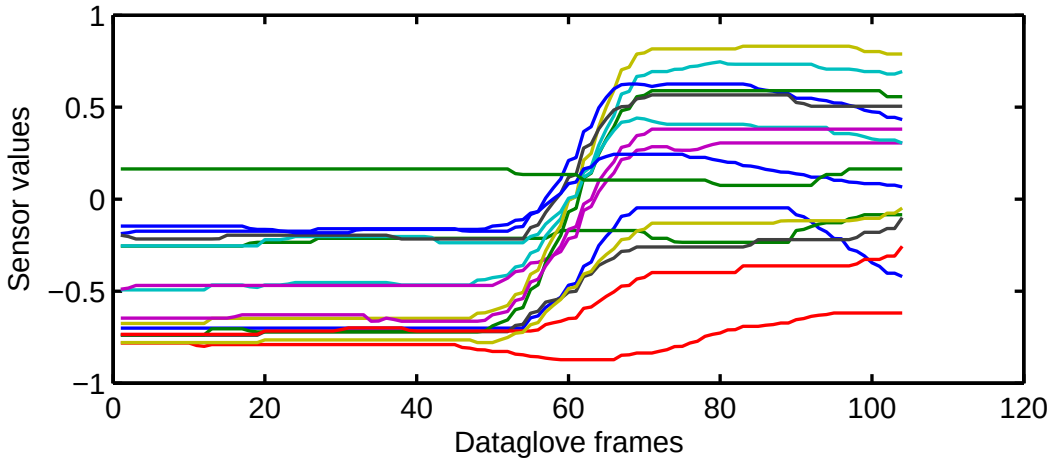


(b) *Another, 'stretched-out' instance of the demaximise gesture.*

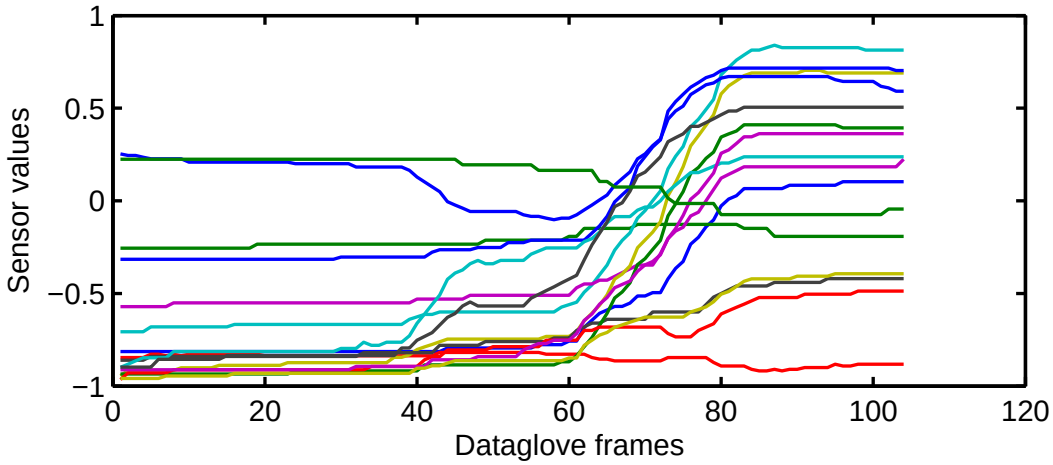
Figure 5.1.: *Examples of the demaximise gesture showing two different instances by the same user. For reasons of clarity, only the glove data is shown.*

Interpolation is used to ensure that gesture instances have the same length. This is done byeither reducing or increasing the length of a gesture to make it equal to the mean length of all gesture instances in the data set [33, 71]. Reducing the number of samples in a gesture is done by removing samples evenly

distributed across the entire instance. Increasing the number of samples is done by evenly adding samples between the existing ones through linear interpolation. Figure 5.2 shows the interpolated variants of the gestures from Figure 5.2. The patterns of the interpolated gestures are now more alike. The recognition results obtained using interpolation are shown in Table 5.4.



(a) *The interpolated variant of Figure 5.1a.*



(b) *The interpolated variant of Figure 5.1b.*

Figure 5.2.: *Two interpolated instances of the demaximise gesture from the same user.*

Technology	Recognition rate using unprocessed data	Recognition rate using interpolated data
CHMM	35.31%	39.58%
DHMM	43.40%	48.27%
LDCRF	92.56%	90.61%

Table 5.4.: *Recognition rates based on default parameter settings using interpolated data. The rates of Table 5.1 are shown for comparison.*

Compared to the experiments using unprocessed data, both CHMMs and DHMMs improved considerably when interpolation was used. However, with LDCRFs the recognition rate decreased slightly.

5.2.4. Dimension reduction through Principal Component Analysis

Training a recogniser becomes increasingly more complex as the number of dimensions of the input vector (i.e. the variables to be measured) increases. The input feature vector has 23 dimensions as described in Chapter 3. A comparison with previous research in glove-based hand gesture recognition shows that this is on the high side. Murakami et al. [59] use a sixteen-dimensional input vector to train an Artificial Neural Network for dynamic gesture recognition. Jong-Sung et al. [37] also use sixteen-dimensional input for single-handed sign language recognition. Fewer features are also used in visual hand gesture recognition. In order to recognise dynamic, bimanual gestures based on strokes Shamaie et al. [74] track the x, y-velocities of both hands, resulting in a four-dimensional input vector to recognise bimanual dynamic gestures. Just et al. [31] applied three-dimensional position and difference features to the detection of bimanual gestures. This resulted in a feature vector of six dimensions for a single hand. The corresponding recognition rates lie between 70% and 99%.

One of the more commonly used techniques to reduce complexity is Principal Component Analysis (PCA) [75, 64]. The goal of PCA is to reduce the dimensions by showing how much information each feature from a given set of features contributes to the total. In other words, the features that have the largest variance are the most descriptive for determining differences between data sets. Features that have little variance are removed, resulting in a reduced input feature vector.

In the experiments reduction was performed through PCA by calculating the percentage of each feature's variance in respect to the total. For the purposes of this research the threshold for removing a feature was set to 1.5% (i.e. if it

contributes less than 1.5% to the total variance). This value was chosen because it proved to reduce a considerable amount of dimensions during initial experiments without losing too much variance. A PCA on unprocessed data shows that the 23 dimensions are reduced to eight, which still retain 95% variance. Figure 5.3 shows a scree graph of the principal component variances. Such a plot often shows a clear separation between the fraction of total variance where the 'most important' components cease and the 'least important' components begin. Such a point is clearly seen between principal components one and two. However, principal component one only explains less than 50% of the total variance. Hence, more components were used in the reduction. The effects of this reduction on the recognition rate are shown in Table 5.5.

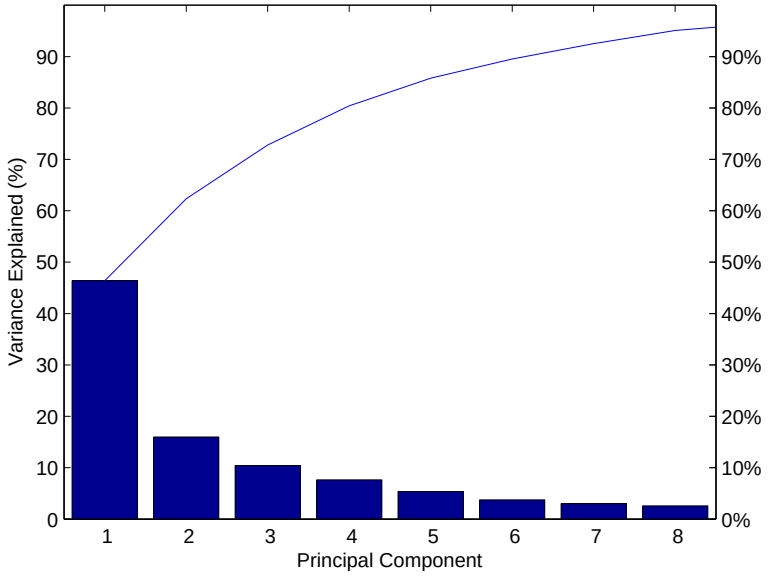


Figure 5.3.: *Scree graph of PCA on the data set with 23 dimensions. For each principal component it is shown how much it explains the data (percentage of variability). The solid line summates the percentage of variability across the principal component axis.*

Technology	Recognition rate using unprocessed data	Recognition rate using data with PCA applied
CHMM	35.31%	22.68%
DHMM	43.40%	32.74%
LDCRF	92.56%	84.34%

Table 5.5.: *Recognition rates based on default parameter settings using data with fewer dimensions through PCA. The rates of Table 5.1 are shown for comparison.*

All three recognition technologies did not benefit from the dimension reduction. On the contrary, all recognition rates decreased significantly when compared to the experiments using unprocessed data. This was unexpected because only as little as 5% of the variance from the total data set was removed as a consequence of the dimension reduction. A possible explanation could be that the data was not in the same units prior to doing the PCA (the position / orientation data differs from the glove data) [53]. Better results might be obtained by combining this method with rescale or min/max normalisation. In Section 5.2.6 on combining pre-processing methods, the results of this experiment are described.

5.2.5. Adding the position difference feature

According to previous research [31, 36], using the difference between two positions as a feature proved useful in recognising hand gestures. The aim in this research is that this new feature adds significantly to the determination of differences between data sets. As a consequence, a significant positive effect on the recognition rate should be observed. The difference in the 3D position data between each consecutive sample was calculated and added at the end of the 23 dimension feature vector. As described in the literature, the first observation vector of each observation sequence was discarded after having added the difference feature of the first and second sample to the feature vector of the second sample. The difference between the second and third sample was added to the third, the difference between third and fourth to the fourth and so on. The difference between two positions was multiplied by 100 to obtain values within the same order of magnitude as those of the 3D position.

PCA was used to find out if the new feature added important information to the total feature vector. A comparison of the results of the PCA with those in Figure 5.3 shows that the eight dimensions that contribute the most value in both cases were not significantly different. In other words, the position difference feature is not of value in this recognition process. The results of the experiments shown in Table 5.6 confirmed this conclusion. The recognition rates remained virtually the same or decreased due to the added complexity of the extra dimensions.

Technology	Recognition rate using unprocessed data	Recognition rate using data with position difference
CHMM	35.31%	35.50%
DHMM	43.40%	40.11%
LDCRF	92.56%	90.02%

Table 5.6.: *Recognition rates based on default parameter settings and with the extracted position difference feature added to the feature vector. The rates of Table 5.1 are shown for comparison.*

5.2.6. Combined methods

The combination of pre-processing methods may lead to a greater improvement in recognition rates when compared to the individual methods. The previous subsections showed that the rescale, normalisation and interpolation pre-processing methods were effective with regard to recognition based on HMMs. The recognition rates increased considerably when compared to the results obtained using unprocessed data. However, the new position difference feature showed no effect and reducing the dimensions through PCA showed a decrease with regard to all recognisers. With LDCRF only a slight increase was observed when rescaling was applied to the data. The other methods either had no effect on the recognition rate or decreased it.

Combinations involving PCA reduction, rescale and min/max normalisation are also of interest because of the fact that PCA might give better results when all data is in the same unit (as explained in Section 5.2.4).

Considering these facts, combining pre-processing methods is only useful for the HMM recognisers. Table 5.7 shows the pre-processing method combinations and the resulting recognition rates.

Combinations	Recognition rate CHMMs	Recognition rate DHMMs	Recognition rate LDCRFs
Rescale - Min/Max normalisation	42.60%	44.71%	91.00%
Rescale - PCA reduction	41.55% (-6.84%)	42.60% (-6.81%)	84.93% (-6.76%)
Min/Max normalisation - PCA reduction	41.68% (-7.62%)	40.57% (-7.49%)	86.30% (-6.52%)
Min/Max normalisation - Interpolation	52.53%	54.14%	N.A.
Rescale - Interpolation	52.66%	49.90%	N.A.
Rescale - Min/Max normalisation - PCA reduction	39.25% (-7.66%)	41.62% (-7.56%)	86.11% (-6.52%)
Rescale - Min/Max normalisation - Interpolation	56.08%	54.63%	N.A.
Rescale - Min/Max normalisation - PCA reduction - Interpolation	53.78% (-7.61%)	54.44% (-7.58%)	N.A.

Table 5.7.: *Recognition results using pre-processing method combinations. PCA reduction removed 12 dimensions. The value in brackets indicates how much information was lost in the process. N.A. indicates that the combination was not considered useful to be tested for the recogniser in question.*

Comparison of the results of the method combination listed first 5.7 to those in tables 5.2 (rescale only) and 5.3 (min/max normalisation only), shows that there is no improvement. The recognition rates obtained using this method combination are similar to those obtained using rescale and min/max normalisation respectively. With regard to CHMMs this method combination performs slightly less well (approximately 1%) when compared to the results using only rescale. With regard to DHMMs this method combination performs slightly less well (approximately 2%) when compared to the results using min/max normalisation only.

The second method combination listed, PCA reduction on rescaled data ensures that HMMs performs significantly better than PCA reduction on unprocessed data (Table 5.5). The rate of CHMM recognition was increased by about 19% and for DHMMs by about 10%, but no significant effect was observed with regard to LDCRFs. This occurred with a minimal loss of variance (< 7%) and a reduction of twelve dimensions.

Combining min/max normalisation with PCA reduction had with regard to CHMMs, an equal effect on the recognition rate (19%), with regard to DHMMs a less positive effect (~8%) and with regard to LDCRFs a positive effect (~2%) compared to the previous method combination.

The method combination listed in row four of the table shows considerable increases in the recognition rates of all but the LDCRFs recogniser.
The combination of rescale and interpolation performs significantly better than the two separately. Wehn compared to rescale (Table 5.2) the recognition rates were increased by approximately 9% and 6% for CHMMs and DHMMs respectively. When compared to interpolation (Table 5.4) the recognition rates were increased by about 13% and 2% for CHMMs and DHMMs respectively.
Adding PCA reduction to the rescale and min/max normalisation method combination (see sixth row of Table 5.7) had a negative effect on the recognition rates of the HMMs. For both CHMMs and DHMMs the recognition rate decreased by about 3%. The recognition rate of the LDCRFs was not significantly affected. When compared to PCA reduction on rescaled data, DHMMs performed less well by approximately 1% and CHMMs also by about 2%.
The rescale, min/max normalisation and interpolation method combination resulted in considerable increases in the recognition rates for both CHMMs and DHMMs. When compared to the rescale and min/max normalisation method combination the recognition rate increases were about 13% (CHMMs) and 9% (DHMMs) and when compared to the rescale and interpolation method combination approximately 3% (CHMMs) and 4% (DHMMs). For CHMMs, this method combination also performed better by about 4% when compared to the min/max normalisation and interpolation method combination. DHMMs performed virtually the same.
For the method combination listed in the last row PCA reduction was added to rescale, min/max normalisation and interpolation. When compared to the previous method combination, PCA reduction reduced 12 dimensions for both CHMMs and DHMMs, but at the cost of about 2% in recognition for CHMMs and under one percent for DHMMs. However, when compared to the other method combinations this method combination performed second best and has the advantage of reducing complexity.

5.3. Choosing parameters

This section describes the results of the experiments in which the parameters of all recognisers were tuned. First the Discrete HMMs were tuned, as described in Subsection 5.3.1, then the Continuous HMMs (Subsection 5.3.2) and lastly the Latent-Dynamic Conditional Random Fields (Subsection 5.3.3).

5.3.1. Discrete HMMs

Combining the rescale, min/max normalisation and interpolation pre-processing methods gave the best recognition results with regard to DHMMs (54.63%, Table 5.7). Because of the numerical instability described in Subsection 4.1.3 the HMM training algorithms were reimplemented and the data was also normalised to the interval $[0, 2]$. The tuning experiments described in this subsection were run on the renormalised data set with interpolation applied.

In Subsection 4.1.1 the number of symbols was described as an additional parameter to the number of states. The results on finding the interval with the best recognition rates are shown in Figure 5.4.

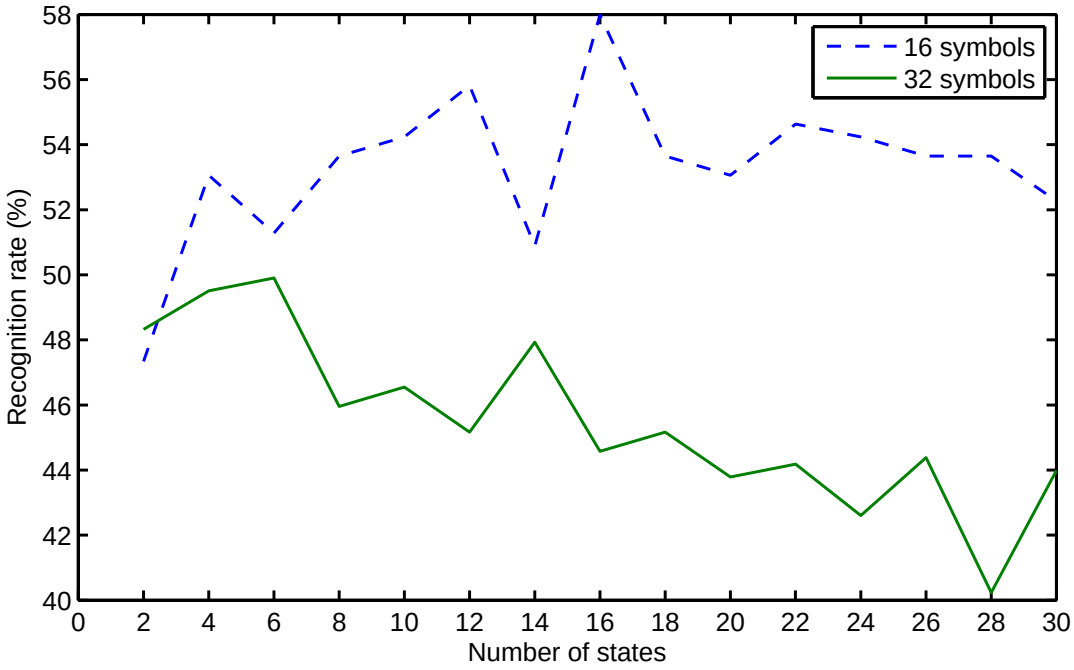


Figure 5.4.: *Recognition rates per 16 and 32 number of symbols for each* $2m$ *states where* $m = 1...15$.

It shows two lines which represent the recognition rates of 16 and 32 symbols. The best recognition rates were obtained with 16 symbols. The highest being 57.99% with 16 states. Because DHMMs with 32 symbols performed significantly worse than with 16 symbols, tests with a higher number of symbols were not considered useful.

Through tuning in the range of 8 to 18 symbols the optimal settings for the number of symbols and states were obtained. Figure 5.5 shows the results. The

setting of 26 states and 8 symbols gave the best result with a recognition rate of 65.48%.

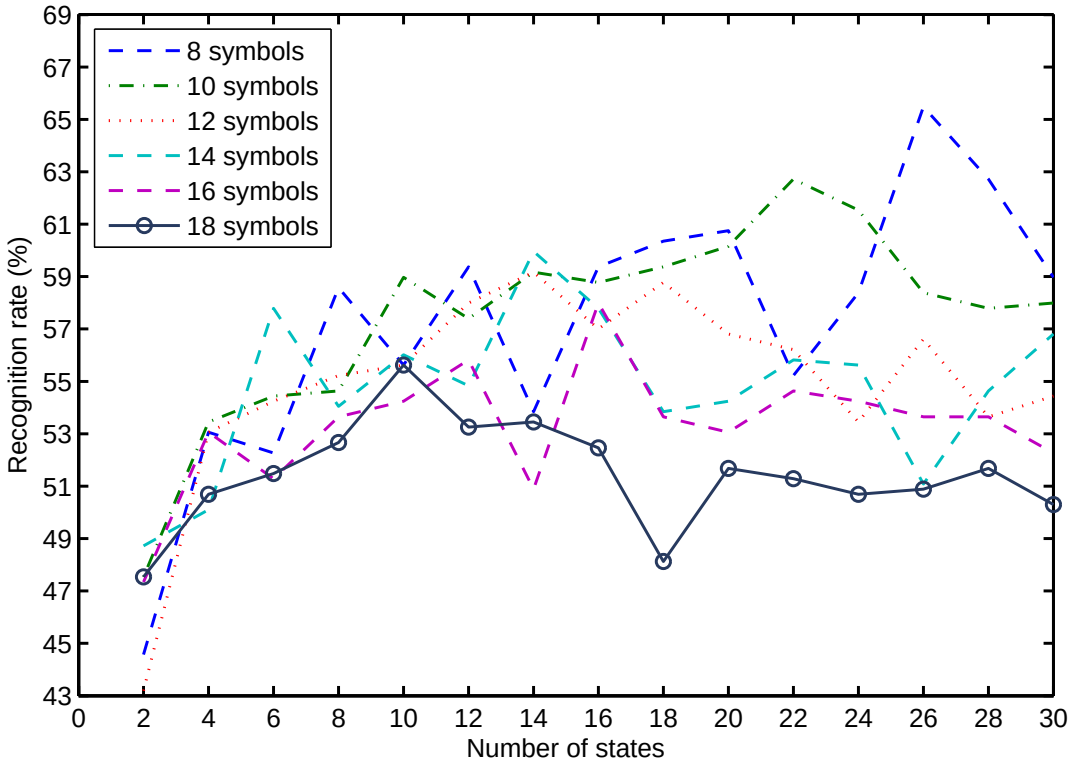


Figure 5.5.: *Recognition rates of the final tuning experiments per* $8 + 2n$ *number of symbols for all* $2m$ *states where* $n = 0...4$ *and* $m = 1...15$*. Best recognition rate: 65.48% (26 states, 8 symbols).*

In Subsection 4.1.2 a third parameter for tuning was described: the hidden state architecture. Throughout the previous experiments the left-to-right architecture was used because within a left-to-right algorithm the state number progresses as time progresses. Intuitively, this closely resembles the hand gesture process. More closely than with an ergodic architecture, as with this architecture it is possible to jump back and forth from any state to any other state. Therefore, training the DHMMs with an ergodic architecture should intuitively result in a lower recognition rate. The DHMMs was trained and tested with this architecture on the best state and symbol setting from Figure 5.5. The average recognition rate after training and testing three times was 63.58%. Only slightly lower than with a left-to-right architecture, which is explained by the use of random training and test sets.

5.3.2. Continuous HMMs

For recognition with CHMMs the best combination of pre-processing methods was: rescale, min/max normalisation and interpolation (56.08%, Table 5.7). As with DHMMs, the HMM training algorithms were reimplemented as described in Subsection 4.1.3 and the data was also normalised to the interval $[0, 2]$. The tuning experiments described in this subsection were run on the renormalised data set with interpolation applied.

The first tuning experiments were done to determine the most optimal parameter settings for CHMMs. Unfortunately, it was not possible to run experiments as far as we wanted. Taking the logarithms of probabilities did make the training of CHMMs numerically stable, but it also made it less efficient. Because the duration of an experiment multiplied by about three, it was only possible to run the experiments for 2, 4, 6 and 8 Gaussians. Figure 5.6 shows the results.

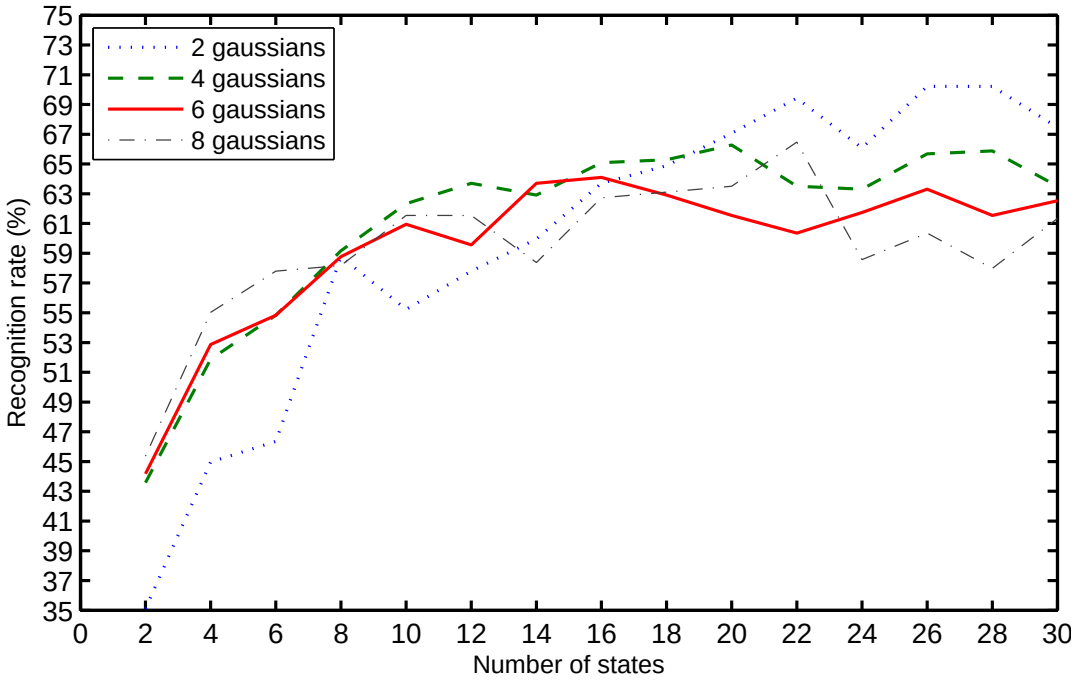


Figure 5.6.: *CHMM recognition rates using logarithms of probabilities.*

The best recognition rate in these experiments was obtained using 26 states and 2 Gaussians: 70.22%.

Using these settings the effects of changing the state architecture parameter were studied. All the results in this subsection were obtained with the left-to-right architecture. Three experiments were then run with the ergodic state

architecture using the parameters that gave the best recognition rate (i.e. 26 states and 2 Gaussians). The average recognition rate was 71.04%.

5.3.3. Latent-Dynamic Conditional Random Fields

With regard to LDCRFs no pre-processing method had a positive effect on the recognition rate, with the exception of rescaling. Therefore, the tuning experiments were all executed with rescaled data. The parameters of LDCRFs were described in Section 4.2. This subsection shows the results of the experiments to tune these parameters. When tuning one parameter, the other parameters kept their default values as described in Section 5.1.

The first parameter that was tuned was the number of hidden states. Because a satisfying recognition rate had already been obtained using one hidden state, increasing the number of hidden states was not expected to have much effect. Table 5.8 shows the results and confirms this expectation. The recognition rate does not increase.

Number of states	Recognition rate
1	93.35%
2	90.61%
3	93.15%
4	92.96%
5	91.59%

Table 5.8.: *Results of the experiments on tuning the LDCRF hidden state parameter.*

The second parameter that was tuned was the regularisation factor. Experiments using this parameter were executed with values ranging from 10^{-3} to 0 and from 1 to 10^{3}, which were taken from Morency et al. [57]. The results of the experiments are shown in Table 5.9.

Regularisation factor	Recognition rate
0.001	40.70%
0.01	46.77%
0.1	86.69%
1	91.78%
10	93.35%
100	90.41%
1000	89.43%

Table 5.9.: *Results of the experiments on tuning the LDCRF regularisation factor parameter.*

The recognition rate increased and reached its maximum in this table at a regularisation factor of 10. It then seemed to decrease, but a higher maximum could possibly be obtained by running experiments with a regularization factor between 1 and 100. However, this would require more timeconsuming tests, for which there was no time in the current research.

The last LDCRF parameter that was studied was the window size parameter. Table 5.10 shows the results.

Window size	Recognition rate	Window size	Recognition rate
0	93.35%	4	94.91%
1	94.52%	5	93.93%
2	94.32%	6	93.15%
3	95.10%		

Table 5.10.: *Results of the experiments on tuning the LDCRF window size parameter.*

It shows a slight upward trend in the recognition rate with a maximum at a window size of 3 and it then shows a downward trend. Therefore, taking a window size of 3 is for the purposes of this research the most beneficial to the recognition rate.

5.4. Quantity of available user data

To see how the quantity of available user data affects the performance, eight experiments were run. Only experiments with LDCRFs were run, considering the fact that the highest recognition rates were achieved with LDCRFs (see Section 5.3). Table 5.11 shows the recognition rates for the increasing number of users using unprocessed data with untuned parameter settings. As the table shows, the increasing quantity of available user data has a significant positive effect on the performance. After six users it seems that most of the variation in gestures has been seen in this case, because the seventh and eighth user do not add to the recognition rate.

Number of users	Recognition rate	Number of users	Recognition rate
1	85.19%	5	91.60%
2	86.41%	6	92.56%
3	90.03%	7	92.53%
4	90.41%	8	91.82%

Table 5.11.: *The recognition rates of LDCRFs using an increasing quantity of user data. Rates are based on unprocessed data and using untuned parameter settings.*

5.5. User independence

The ultimate goal of this thesis is to achieve a recognition percentage that is usable for a large group of users. Because training data is not available from all potential users, the recogniser should generalise over unseen users. It should satisfactorily recognise gestures independent of its user. This section shows the results that were obtained while testing our recogniser for user independence. Subsection 5.5.1 describes the method of testing used. Subsection 5.5.2 gives a description of the additional data that was prepared for the tests and 5.5.3 shows the results of the tests.

5.5.1. Method

As subject for our tests the LDCRF recogniser was chosen because it performed the best in the previous experiments. In the tests in the previous sections the training set contained randomly selected data of all users available. The test set contained the remaining data of all users. In the tests in this section the training set contained all data from a particular number of users and the test set all data of the remaining users. The difference being that the recogniser is now tested on users not seen during the training process at all. To create a more similar setting to that of the tests in the previous sections, the LDCRF was trained on the six already available users. As test set data from two additional users was segmented and labelled (see Subsection 5.5.2). To research the user independent behaviour of our recogniser, tests were executed with test sets containing the data of user 1 and user 2 separately. Tests were also run in which User 1 was added to the training set and User 2 to the test set and vice versa. This way, the effect of more training data on user independence was studied.

5.5.2. Data

Data of two additional users was cleaned and segmented for the user independence tests. Both users are male, between 20 to 25 years old and right-handed. User 1 has a bachelor's degree in computer science and has some experience with gesture interfaces through the use of touch screens and the Wii gaming console. User 2 was doing his bachelor thesis in Art & Technology at the time of recording. He has additional experience with gesture interfaces through multi-touch surfaces. Table 5.12 shows the number of samples used in the tests for both users.

Class	Close	Demax.	Demin.	Max.	Min.	Point	Select	Zoom in	Zoom out
User 1	22	13	24	21	23	274	87	6	8
User 2	26	20	24	24	24	238	129	39	14
Total	48	33	48	45	47	578	217	45	22

Table 5.12.: *Number of gesture samples per class from the two additional users.*

Just as in the previous sections, the data with which the recogniser was trained and tested was rescaled to the $[-1, 1]$ interval.

5.5.3. Results

This subsection shows the test results. The recogniser was trained using the parameter settings which obtained the best results in Subsection 5.3.3: 1 hidden state, regularisation factor of 10 and a window size of 3. Four tests were run using the data sets of the additional users, the results of which are shown in Table 5.13.

Test	Recognition rate
6 users training, test with User 1	91.21%
6 users training, test with User 2	90.71%
6 users + User 1 training, test with User 2	91.64%
6 users + User 2 training, test with User 1	94.77%

Table 5.13.: *Recognition rates obtained with the tests on user independence. '6 users' is taken to mean the six already available users and 'User 1' and 'User 2' are taken to mean the two additional users from whom the data was segmented and labelled especially for these tests.*

The recognition rates of the tests listed in the first two rows are lower compared to the 93.35% of Table 5.2. In those tests the results were obtained by training the LDCRFs on the gesture styles of all users with independent training and test sets. However in this section all styles but one were used for training purposes, the one that was not formed the test set. This explains the lower recognition rates. When you look at their confusion matrices shown in tables 5.14 and 5.15, it becomes clear that the Point gesture instances were mainly misrecognised. Furthermore, the Select gesture instances were also frequently misrecognised according to Table 5.14.

Class	Close	Demax.	Demin.	Max.	Min.	Point	Select	Zoom in	Zoom out	Total
1	21	0	0	0	1	0	0	0	0	22
2	1	12	0	0	0	0	0	0	0	13
3	0	0	24	0	0	0	0	0	0	24
4	0	0	0	21	0	0	0	0	0	21
5	0	0	0	0	23	0	0	0	0	23
6	0	0	0	0	0	251	10	2	11	274
7	0	0	0	0	0	13	74	0	0	87
8	0	0	0	0	0	1	0	2	3	6
9	0	0	0	0	0	0	0	0	8	8

Table 5.14.: *Confusion matrix of the results obtained using 6 users as training set and User 1 as test set. Recognition rate: 91.21%*

Class	Close	Demax.	Demin.	Max.	Min.	Point	Select	Zoom in	Zoom out	Total
1	22	0	0	0	4	0	0	0	0	26
2	0	20	0	0	0	0	0	0	0	20
3	0	0	24	0	0	0	0	0	0	24
4	0	0	0	24	0	0	0	0	0	24
5	0	0	0	0	24	0	0	0	0	24
6	0	0	0	0	0	196	10	2	30	238
7	0	0	0	0	0	0	129	0	0	129
8	0	0	0	0	0	0	0	39	0	39
9	0	0	0	0	0	0	0	4	10	14

Table 5.15.: *Confusion matrix of the results obtained using 6 users as training set and User 2 as test set. Recognition rate: 90.71%.*

The misrecognition in both tables is explained by the great difference in gesture style between the training and test users in combination with the definition of the gestures. The Point and Select gestures are similar to each other, because a Select gesture starts and ends with a Point pose of the hand. Zoom in and out are similar because they only differ in the way the thumb and index finger move (either toward or from each other). These similarities make it difficult for a recogniser to discriminate between either the Point or Select gestures and the Zoom in or out gestures. In addition, Point gestures were also misrecognised as Zoom in or out gestures. A possible explanation could be the fact that some of the Point instances looked similar to the starting pose of the Zoom gestures. This was however, not confirmed.

The results of the tests listed in the third and fourth row of Table 5.13 show an increase in recognition rate compared to the results listed in the first two rows. These increases are explained by a better recognition of the Point gesture

as tables 5.16 and 5.17 show. However, the recognition of Zoom out gestures decreased slightly.

Class	Close	Demax.	Demin.	Max.	Min.	Point	Select	Zoom in	Zoom out	Total
1	19	2	0	1	0	0	0	0	0	22
2	0	13	0	0	0	0	0	0	0	13
3	0	0	24	0	0	0	0	0	0	24
4	0	0	0	21	0	0	0	0	0	21
5	0	0	0	0	23	0	0	0	0	23
6	0	0	0	0	0	272	2	0	0	274
7	0	0	0	0	0	14	73	0	0	87
8	0	0	0	0	0	0	0	5	1	6
9	0	0	0	0	0	0	0	5	3	8

Table 5.16.: *Confusion matrix of the results obtained using 7 users as training set and User 2 as test set. Recognition rate: 91.64%*

Class	Close	Demax.	Demin.	Max.	Min.	Point	Select	Zoom in	Zoom out	Total
1	24	0	0	0	2	0	0	0	0	26
2	0	20	0	0	0	0	0	0	0	20
3	0	0	24	0	0	0	0	0	0	24
4	0	1	0	23	0	0	0	0	0	24
5	0	0	0	0	24	0	0	0	0	24
6	0	0	0	0	0	202	5	6	25	238
7	0	0	0	0	0	1	128	0	0	129
8	0	0	0	0	0	0	0	39	0	39
9	0	0	0	0	0	0	0	5	9	14

Table 5.17.: *Confusion matrix of the results obtained using 7 users as training set and User 1 as test set. Recognition rate: 94.77%.*

Chapter 6

Conclusion

Considering the results from Chapter 5, let us first look at the initial research questions and answer them:

1. Can dynamic hand gestures be recognised independent of users?
   a) What technologies are suited for this purpose?
      i. What effect does pre-processing the data set have on the performance?
      ii. What effect does the tuning of parameters have on the performance?
      iii. How does a traditional technology compare to a novel one?
      iv. What effect does the quantity of available user data have on the performance?
   b) Can a recogniser be made user independent for a large group?

The main research question (1) is answered by this thesis as a whole. It details one approach to user-independent recognition of dynamic hand gestures using a dataglove and a position and orientation sensor. The sensors recorded training data in a Wizard of Oz experiment. The recorded data was manually segmented and then used to train and test recognisers.

The effects of pre-processing on the performance varied per recogniser (1a, i). Both rescaling and min/max normalising the data to the $[-1, 1]$ interval (Subsections 5.2.1 and 5.2.2) had a significant positive effect on the recognition rate of CHMMs, because both pre-processing methods improved the visibility of the patterns in the data. DHMMs and LDCRFs were not significantly affected by these two methods. For DHMMs this could be due to the vector quantisation

process that minimises the effects prior to the actual training. For LDCRFs it could be that these two methods did not make the distinct differences between gesture classes stand out compared to the other data.
Interpolating the data had a large positive effect on both HMMs (see Subsection 5.2.3). Gestures within a class were more alike in the case of HMMs when the gesture instances were interpolated to the same length. For LDCRFs, interpolation could possibly have removed discriminating features, which could explain the small negative effect.
Dimension reduction through Principal Component Analysis (PCA) when applied to unprocessed data had a significant negative effect on all recognisers (see Subsection 5.2.4). After applying it to standardised data it still had a negative effect, albeit a smaller one. It is possible that PCA removed important dimensions which all recognisers used to discriminate gestures. However, a considerable reduction in training time was observed.
The introduction of the position difference feature (see Subsection 5.2.5) had no significant effect on the recognition with CHMMs and LDCRFs. This can be explained by comparing the PCA of the data with this feature to the PCA of the data without this feature (Figure 5.3). The comparison showed no significant increase in the variance of the principal components. The negative effect on DHMMs could possibly be explained by the added complexity to the vector quantisation process due to the extra dimensions.
With regard to the pre-processing methods, it can be concluded that the method combination of rescale, min/max normalisation and interpolation gave the best results for both HMM technologies (CHMM: 56.08%, DHMM: 54.63%, Table 5.7). For LDCRFs rescaling on its own gave the best results (93.35%, Table 5.2).

The effects of tuning the recogniser parameters were generally beneficial to the recognition rate (1a, ii). The numerical overflows encountered during tuning were solved with a numerically stable reimplementation of the HMM algorithms (see Subsection 4.1.3). DHMMs were modelled best with 8 symbols, 26 states and the left-to-right architecture (see Subsection 5.3.1). CHMMs were modelled best with 2 Gaussians, 26 states and again the left-to-right architecture (see Subsection 5.3.2). With a larger number of symbols and Gaussians, the increase in recognition rate levelled off sooner as the number of states increased. For both HMMs the number of states were equal. This confirmed that the internal substructure of the gestures had been found, despite the low recognition rates. LDCRFs performed best with 1 hidden state, a regularisation factor of 10 and a window size of 3. The hidden state parameter had no significant effect on the

recognition rate (see Table 5.8). The question is whether LDCRFs can find any internal substructure within the gestures. The regularisation factor improved the recognition rate in an asymptotic manner (see Table 5.9). The window size parameter improved the recognition rate slightly when varied from 0 to 3 (see Table 5.10).

The traditional CHMM and DHMM technologies do not compare to LDCRFs in this case (1a, iii). From the beginning LDCRFs have significantly outperformed both CHMMs and DHMMs. Having applied pre-processing methods and tuned the recogniser parameters, LDCRFs still outperformed both, providing satisfactory recognition rates (>=93.35%), whereas CHMMs (70.22%) and DHMMs (66.10%) did not. Unfortunately, this could not be explained, but it is assumed that the different nature of the models is what is causing this, as HMM is generative and LDCRF in contrast is discriminative. The advantage of the latter is that it does not aim to fully model a data distribution, which is more complex. While comparing the traditional technologies it was found that CHMMs outperformed DHMMs by about 5%, most likely because the DHMMs discretise the continuous observations during which important information is lost.

With LDCRFs experiments were run to study the effect of the quantity of user data on the performance (1a, iv). It can be concluded that a larger quantity of user data can have a positive effect on the performance. However, the experiments also indicate that there is probably a point where adding new data does not outweigh the cost of training.

In almost all user-independence tests, LDCRFs achieved satisfactory recognition rates of over 91%. Although these results were achieved with test sets of only two users, the researcher is confident that LDCRFs are suitable for user-independent dynamic gesture recognition (1b).

In the course of this research into recognising user-independent dynamic hand gestures many aspects relating to the technology behind the recognisers were discovered. For example, how data pre-processing and parameter tuning affect them. Knowledge was gained on how to apply certain technologies to best achieve the goal. However, there is more to hand gesture interaction. Issues including how users will receive such a technology and if and how they will use it in completing their tasks have not been studied. The approach used provided over 91% recognition independent of users, which means that about 1 in 10 user gestures is not recognised correctly. The recognition of a particular gesture is often connected to a certain interface action, which when applied to this research means that an unintended interface action is executed with every mis-

recognition. This could be disastrous when, for example, important information is thereby accidentally deleted. To prevent such misrecognitions, variants of likelihood thresholding are proposed by [45, 34]. Zobl et al. [100] propose using confidence measures. In addition, human errors could also be made (i.e. making the wrong gesture for a certain action). It would be interesting to find out if users could easily recover from such errors and how. Another question that is of interest is whether 91% recognition of gestures suffices for usable interface interaction. Do users mind remaking 1 in 10 gestures, for example? Both these issues and the technological aspect of gesture recognition are important.

This research has only studied hand gesture interaction in part. Nevertheless, it can be concluded that the recognition of user-independent dynamic hand gestures as described in this thesis is not only feasible, but it can without a doubt also play an effective role in the successful realisation of hand gesture interaction.

Chapter 7

Future research

This research has answered the question whether the user-independent recognition of dynamic hand gestures is feasible. One of the recognition technologies that was applied has shown satisfying results. The following paragraphs describe several ideas for future research.

Continuous Hidden Markov Models (CHMMs) and Discrete Hidden Markov Models (DHMMs) performed poorly in the tests when compared to Latent-Dynamic Conditional Random Fields (LDCRFs). This was unexpected considering the fact that previous research had shown higher recognition rates [98, 34]. The training and test data were taken as a starting point in order to try and find an explanation. Since numerical overflow was experienced even when scaling was applied, it is possible that the observation sequences were too long. In addition, our data was recorded at a frame rate of 70 fps. Vision-based recognition commonly employs a frame rate of only 30 fps. The observation sequences used in this research are thus 2.33 times longer. Future research could resample the data set to 30 fps and study the recognition rate with shorter observation sequences. Another explanation can be found in the fact that the same parameter settings were used for each gesture class. Given that the gestures are not the same, a different number of states, Gaussians or symbols per gesture could possibly give a better fit on the data. Future research could provide insight into the effect varying the parameter settings depending on gesture class has on the recognition rate. The state architecture parameter could possibly increase the performance of the HMMs. A left-to-right hidden state architecture was used with only two transitions. Adding transitions between more states could possibly improve the recognition rate, because the HMM can then 'skip' states, if

dealing with a short gesture.
Even though benchmark tests for the DHMM algorithms used had shown that the extension of multi-labelling improved recognition, this might not be the case for the real-world context of this research. In future research this could be tested using DHMM algorithms without the extension.

Before expanding this research with LDCRFs to new topics, future research should first study why varying the hidden state parameter had no significant effect on the recognition rate. By running the same tests with the 'ancestral' CRF and HCRF algorithms it would become clear whether the extensions of LDCRF add to the recognition of gestures. Furthermore more tests could be done with regard to user independence. By employing cross validation tests on the current data set the claims on user independence could be reinforced.
More gesture data was recorded than was used in the end. This data could possibly still be used in training the LDCRFs for unseen variation among the gesture classes. To find out if this data contributes to the variation, future research could establish a method for determining whether a new data set contains unseen variation. This method could provide a measure for making the decision to add data from an unseen user for later offline retraining.
During this research the HMM algorithms used were adapted for parallel processing, which greatly reduced the training time on multicore commodity PCs. Future research could study the potential of parallel processing in general. In addition, the HCRF library could also be adapted for parallel processing. Advances have also been made in the use of the chips of graphic cards (GPU) for intensive parallel calculations[1]. GPUs are optimised for multiplication operations for fast 3D graphics rendering. That optimisation could potentially be of use in this research, because the training algorithms also rely heavily on multiplication operations. By using the GPU the CPU could be offloaded to do other tasks.
One of the directions in which to expand this research with LDCRFs is the development of a real-time recognition system. In the long run such a system could be used to design a hand gesture interface. One of the most important topics that would need to be covered for such a system is the automatic segmentation of gestures from a continuous data stream. LDCRFs are also suited to this task, because they model both the dynamics within and between gesture classes [57]. They classify each observation in a sequence of observations, which gives a starting point for discriminating the beginning and end of a gesture in a stream.

[1] See http://gp-you.org. Last visited on October 5, 2010.

As the above paragraphs show, there are still many avenues to explore. However, with this thesis, a firm foundation for future research further exploring these avenues has been laid.

Bibliography

[1] E. Alpaydin. *Introduction to Machine Learning*. The MIT Press, MA, USA, 1st edition, 2004.

[2] Ascension Technology Corporation, Burlington, VT, USA. *The Flock of Birds Installation and Operation Guide*, March 2004. Version 910002-A Rev C. Last retrieved on July 27, 2010 from `ftp://ftp.ascension-tech.com/MANUALS/Flock_of_Birds_Manual-RevC.pdf`.

[3] L.E. Baum and T. Petrie. Statistical inference for probabilistic functions of finte state Markov chains. *Annals of Mathematical Statistics*, 37(6):1554–1563, 1966.

[4] R. A. Bolt. "Put-that-there": Voice and gesture at the graphics interface. In *SIGGRAPH '80: Proceedings of the 7th annual conference on Computer graphics and interactive techniques*, pages 262–270, New York, NY, USA, 1980. ACM.

[5] W. Buxton. A three-state model of graphical input. In *INTERACT '90: Proceedings of the IFIP TC13 Third Interational Conference on Human-Computer Interaction*, pages 449–456, Amsterdam, The Netherlands, 1990. North-Holland Publishing Co.

[6] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 161–168, New York, NY, USA, 2006. ACM.

[7] S. Casale, A. Russo, G. Scebba, and S. Serrano. Speech Emotion Classification Using Machine Learning Algorithms. In *ICSC '08: Proceedings of the 2008 IEEE International Conference on Semantic Computing*, pages 158–165, Washington, DC, USA, 2008. IEEE Computer Society.

[8] D. K. Chaturvedi. *Soft Computing: Techniques and its Applications in Electrical Engineering*. Springer Verlag, Heidelberg, Germany, 2008.

[9] Y. Chen, W. Gao, and M.A. Jiyong. Hand Gesture Recognition Based on Decision Tree. In *Proceedings of the International Symposium on Chinese Spoken Language Processing*. International Speech Communication Association, 2000.

[10] K. Cheng. *Direct interaction with large displays through monocular computer*

vision. PhD thesis, University of Sydney, Sydney, Australia, October 2008. Last retrieved on October 13, 2010 from http://hdl.handle.net/2123/5331.

[11] R. Cutler and M. Turk. View-based Interpretation of Real-time Optical Flow for Gesture Recognition. In *Proceedings of IEEE International Conference on Automatic Face and Gesture Recognition*, pages 416–421, 1998.

[12] N. Dahlbäck, A. Jönsson, and L. Ahrenberg. Wizard of Oz studies: why and how. In *IUI '93: Proceedings of the 1st international conference on Intelligent user interfaces*, pages 193–200, New York, NY, USA, 1993. ACM.

[13] S. David, M.A. Ferrer, C.M. Travieso, and J.B. Alonso. gpdsHMM: a hidden Markov model toolbox in the Matlab environment. In *Complex Systems Intelligence and Modern Technological*, pages 476–479, Cherbourg, France, 2004. Société de l'Electricité, de l'Electronique et des Technologies de l'Information et de la Communication (SEE). Last retrieved on July 18, 2010 from http://www.gpds.ulpgc.es/download/toolbox/gpdshmm/gpdsHMM_vf.pdf.

[14] K.H. Davies, R. Biddulph, and S. Balashek. Automatic Speech Recognition of Spoken Digits. *Journal of the Acoustic Society of America*, 24(6):637–642, 1952.

[15] W. de Leeuw, P. Verschure, and R. van Liere. Visualization and Analysis of Large Data Collections: a Case Study Applied to Confocal Microscopy Data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1251–1258, 2006.

[16] O. Dekel, F. Fischer, and A. D. Procaccia. Incentive compatible regression learning. In *SODA '08: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 884–893, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

[17] D. Demirdjian, T. Ko, and T. Darrell. Untethered gesture acquisition and recognition for virtual world manipulation. *Virtual Reality*, 8(4):222–230, 2005.

[18] P. Dietz and D. Leigh. DiamondTouch: a multi-user touch technology. In *UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 219–226, New York, NY, USA, 2001. ACM.

[19] M. Elmezain, A. Al-Hamadi, J. Appenrodt, and B. Michaelis. A Hidden Markov Model-based continuous gesture recognition system for hand motion trajectory. In *ICPR 2008: 19th International Conference on Pattern Recognition*, pages 1–4, December 2008.

[20] F. W. Fikkert. *Gesture Interaction at a Distance*. PhD thesis, Universiteit Twente, Enschede, The Netherlands, March 2010. Last retrieved on July 14, 2010 from http://eprints.eemcs.utwente.nl/17507/.

[21] F. W. Fikkert, P. van der Vet, H. Rauwerda, T. Breit, and A. Nijholt. Gestures to Intuitively Control Large Displays. pages 199–204. Springer Verlag, Berlin, Germany, 2009.

[22] M. Gales and S. Young. The application of hidden Markov models in speech recog-

nition. In *Foundation and Trends in Signal Processing*, volume 1. now Publishers Inc., Hanover, MA, USA, 2008.

[23] M. Ganzeboom. How hand gestures are recognized using a dataglove. Last retrieved on July 16, 2010 from http://hmi.ewi.utwente.nl/verslagen/capita-selecta/CS-Ganzeboom-Mario-3.pdf, 2009.

[24] S. P. Gill and J. Borchers. Knowledge in co-action: social intelligence in collaborative design activity. *AI & Society*, 17:322–339, 2003.

[25] M. Goudbeek, A. Cutler, and R. Smits. Supervised and unsupervised learning of multidimensionally varying non-native speech categories. *Speech Communication*, 50(2):109–125, 2008.

[26] F. Grize and M. Aminian. Cybcérone: a kiosk information system based on WWW and Java. *ACM Interactions Magazine*, 4(6):62–69, 1997.

[27] J. Y. Han. Low-cost multi-touch sensing through frustrated total internal reflection. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 115–118, New York, NY, USA, 2005. ACM.

[28] K. Hinckley. Input technologies and techniques. In *The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications*, pages 151–168. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 2003.

[29] H. Hörtner, P. Maresch, R. Praxmarer, and C. Naglhofer. Libro Vision: Gesture-Controlled Virtual Book. volume 3105 of *Lecture Notes in Computer Science*, pages 258–263. Springer Verlag, Heidelberg, Germany.

[30] E. L. Hutchins, J. D. Hollan, and D. A. Norman. Direct manipulation interfaces. *Human-Computer Interaction*, 1(4):311–338, 1985.

[31] A. Just and S. Marcel. A comparative study of two state-of-the-art sequence processing techniques for hand gesture recognition. *Computer Vision and Image Understanding*, 113(4):532–543, 2009.

[32] F. Karray, M. Alemzadeh, J. A. Saleh, and M. N. Arab. Human-Computer Interaction: Overview on State of the Art. *International Journal on Smart Sensing and Intelligent Systems*, 1(1):137–159, 2008.

[33] J. Kela, P. Korpipää, J. Mäntyjärvi, S. Kallio, G. Savino, L. Jozzo, and D. Marca. Accelerometer-based gesture control for a design environment. *Personal and Ubiquitous Computing*, 10(5):285–299, 2006.

[34] C. Keskin and L. Akarun. STARS: Sign tracking and recognition system using input-output HMMs. *Pattern Recognition Letters*, 30(12):1086–1095, 2009.

[35] C. Keskin, K. Balci, O. Aran, and L. Akarun. A Multimodal 3D Healthcare Communication System. In *Proceedings of 3DTV-conference* *http://www.3dtv-con.org*, 2007. Last retrieved on July 18, 2010 from http://www.cmpe.boun.edu.tr/~aran/publications/keskin07multimodal.pdf.

[36] I.-C. Kim and S.-I. Chien. Analysis of 3D Hand Trajectory Gestures Using Stroke-

Based Composite Hidden Markov Models. *Applied Intelligence*, 15(2):131–143, 2001.

[37] J.-S. Kim, W. Jang, and Z. Bien. A dynamic gesture recognition system for the Korean sign language (KSL). *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 26(2):354–359, April 1996.

[38] R. Kjeldsen and J. Kender. Toward the use of gesture in traditional user interfaces. In *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition, 1996*, pages 151–156, 14-16 1996.

[39] K. J. Kortbek and K. Gronbaek. Interactive spatial multimedia for communication of art in the physical museum space. In *MM '08: Proceeding of the 16th ACM international conference on Multimedia*, pages 609–618, New York, NY, USA, 2008. ACM.

[40] J. F. Kramer. Determination of thumb position using measurements of abduction and rotation, January 1996. Last retrieved on September 24, 2010 from `http://www.freepatentsonline.com/5482056.html`.

[41] O. Kulyk, I. Wassink, P. E. van der Vet, G. C. van der Veer, and E. M. A. G. van Dijk. Sticks, balls or a ribbon? Results of a formative user study with bioinformaticians. Technical Report TR-CTIT-08-72, Centre for Telematics and Information Technology, University of Twente, Enschede, The Netherlands, December 2008. Last retrieved on 12 Oktober, 2010 from `http://eprints.eemcs.utwente.nl/14622/`.

[42] G. Kurtenbach and B. Buxton. GEdit: a test bed for editing by contiguous gestures. *SIGCHI Bulletin*, 23(2):22–26, 1991.

[43] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[44] C. Lee and Y. Xu. Online, Interactive Learning of Gestures for Human/Robot Interfaces. In *IEEE International Conference on Robotics and Automation*, pages 2982–2987, 1996.

[45] H.-K. Lee and J. H. Kim. An HMM-Based Threshold Model Approach for Gesture Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(10):961–973, 1999.

[46] G. Lin, Y. Fan, and E. Zhang. Human action recognition using latent-dynamic condition random fields. *International Conference on Artificial Intelligence and Computational Intelligence*, 3:147–151, 2009.

[47] Y. Linde, A. Buzo, and R. M. Gray. An Algorithm for Vectore Quantizer Design. *IEEE Transactions on Communications*, 28(1):84–95, January 1980.

[48] J. Liu, L. Zhong, J. Wickramasuriya, and V. Vasudevan. uWave: Accelerometer-

based personalized gesture recognition and its applications. *Pervasive and Mobile Computing*, 5(6):657–675, 2009.

[49] S. Mahmoud. Recognition of writer-independent off-line handwritten Arabic (Indian) numerals using hidden Markov models. *Signal Processing*, 88(4):844–857, 2008.

[50] T. Mandel. *Encyclopedia of Information Systems, Volume Four*, chapter 256. Academic Press, 2002. Last retrieved on July 15, 2010 from http://www.theomandel.com/docs/mandel-encyclopedia.pdf.

[51] T. P. Mann. Numerically Stable Hidden Markov Model Implementation, February 2006. Last retrieved on August 5, 2010 from http://bozeman.genome.washington.edu/compbio/mbt599_2006/hmm_scaling_revised.pdf.

[52] V.-M. Mäntylä. *Discrete hidden Markov models with application to isolated user-dependent hand gesture recognition*. PhD thesis, VTT Electronics, Finland, 2001. Thesis for the degree of Licenciate of Philosophy.

[53] The Mathworks, Inc. *The Matlab User's Guide: Example of Principal Component Analysis - Online version*, 2010. Last retrieved on 3 August, 2010 from http://www.mathworks.com/access/helpdesk/help/toolbox/stats/brkgqnt.html#f14047.

[54] S. Mitra and T. Acharya. Gesture recognition: A survey. *IEEE Transactions on Systems, Man and Cybernetics - Part C: Applications and Reviews*, 37(3):311–324, May 2007.

[55] L.-P. Morency, C. M. Christoudias, and T. Darrell. Recognizing gaze aversion gestures in embodied conversational discourse. In *ICMI '06: Proceedings of the 8th international conference on Multimodal interfaces*, pages 287–294, New York, NY, USA, 2006. ACM.

[56] L.-P. Morency and T. Darrell. Head gesture recognition in intelligent interfaces: the role of context in improving recognition. In *IUI '06: Proceedings of the 11th international conference on Intelligent user interfaces*, pages 32–38, New York, NY, USA, 2006. ACM.

[57] L.-P. Morency, A. Quattoni, and T. Darrell. Latent-Dynamic Discriminative Models for Continuous Gesture Recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1–8, Los Alamitos, CA, USA, 2007. IEEE Computer Society.

[58] P. Morguet and M. Lang. Comparison of approaches to continuous hand gesture recognition for a visual dialog system. In *ICASSP '99: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 3549–3552, Washington, DC, USA, 1999. IEEE Computer Society.

[59] K. Murakami and H. Taguchi. Gesture recognition using recurrent neural networks. In *CHI '91: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 237–242, New York, NY, USA, 1991. ACM.

[60] F. Parvini, D. Mcleod, C. Shahabi, B. Navai, B. Zali, and S. Ghandeharizadeh. An Approach to Glove-Based Gesture Recognition. In *Proceedings of the 13th International Conference on Human-Computer Interaction. Part II*, pages 236–245, Heidelberg, Germany, 2009. Springer Verlag.

[61] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[62] R. Poppe. A survey on vision-based human action recognition. *Image and Vision Computing*, 28(6):976–990, 2010.

[63] L. V. Praseeda and M. Sasikumar. Facial Expression Recognition from Global and a Combination of Local Features. *IETE Technical Review*, 26(1):41–46, 2009.

[64] K. L. Priddy and P. E. Keller. *Artificial Neural Networks: An Introduction*, pages 15–20. The International Society for Optical Engineering, Bellingham, Washington, USA, 2005.

[65] A. Quattoni, M. Collins, and T. Darrell. Conditional random fields for object recognition. In *Advances in Neural Information Processing Systems 17: Proceedings of the 2004 conference*, pages 1097–1104. MIT Press, 2004.

[66] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Readings in speech recognition*, pages 267–296, 1990.

[67] L. R. Rabiner and B. H. Juang. An introduction to hidden Markov models. *IEEE ASSP Magazine*, pages 4–15, January 1986.

[68] B. Ramaprasad and S. Hamori. *Hidden Markov Models: Applications to Financial Economics*. Spriner Verlag, Heidelberg, Germany, 2004.

[69] H. Rauwerda, W.C. de Leeuw, J. Adriaanse, M. Bouwhuis, P. van der Vet, and T.M. Breit. The role of e-BioLabs in a life sciences collaborative working environment. In *Expanding the Knowledge Economy: Issues, Applications, Case Studies*, volume 4, pages 866–873. IOS Press, Amsterdam, The Netherlands, 2007.

[70] B. Raytchev, N. Otsu, and O. Hasegawa. User-independent gesture recognition by relative-motion extraction and discriminant analysis. *New Generation Computing*, 18(2):117–126, 2000.

[71] L. Reng, T. B. Moeslund, and E. Granum. Finding Motion Primitives in Human Body Gestures. In *Gesture in Human-Computer Interaction and Simulation: 6th International Gesture Workshop*, Lecture Notes in Artificial Intelligence, pages 133–144, Heidelberg, Germany, 2006. Springer Verlag.

[72] F. Samaria and S. Young. HMM-based architecture for face identification. *Image and Vision Computing*, 12(8):537–543, 1994.

[73] L. A. Shalabi and Z. Shaaban. Normalization as a Preprocessing Engine for Data Mining and the Approach of Preference Matrix. In *DEPCOS-RELCOMEX '06: Proceedings of the International Conference on Dependability of Computer Systems*, pages 207–214, Washington, DC, USA, 2006. IEEE Computer Society.

[74] A. Shamaie and A. Sutherland. A Dynamic Model for Real-Time Tracking of Hands in Bimanual Movements. In *Gesture-Based Communication in Human-Computer Interaction: 5th International Gesture Workshop*, volume 2915 of *Lecture Notes in Computer Science*, pages 455–456. Springer Verlag, 2004.

[75] J. Shlens. A tutorial on Principal Component Analysis, 2005. Salk Institute for Biological Studies, California, USA. Last retrieved on October 12, 2010 from `http://www.snl.salk.edu/~shlens/pca.pdf`.

[76] B. Shneiderman. Direct Manipulation: A Step Beyond Programming Languages. *IEEE Computer Magazine*, 16(8):57–69, August 1983.

[77] T. Starner and A. Pentland. Real-time American Sign Language recognition from video using hidden Markov models. In *ISCV '95: Proceedings of the International Symposium on Computer Vision*, pages 265–270, Washington, DC, USA, 1995. IEEE Computer Society.

[78] T. Starner, A. Pentland, and J. Weaver. Real-Time American Sign Language Recognition Using Desk and Wearable Computer Based Video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1371–1375, 1998.

[79] H.I. Stern, J.P. Wachs, and Y. Edan. Optimal Consensus Intuitive Hand Gesture Vocabulary Design. In *IEEE International Conference on Semantic Computing*, pages 96–103, 4-7 2008.

[80] D. Stone, C. Jarret, M. Woodroffe, and S. Minocha. *User Interface Design and Evaluation*. Morgan Kaufman, San Francisco, CA, USA, 2005.

[81] X. Sun, L.-P. Morency, D. Okanohara, and J. Tsujii. Modeling latent-dynamic in shallow parsing: a latent conditional model with improved inference. In *COLING '08: Proceedings of the 22nd International Conference on Computational Linguistics*, pages 841–848, Morristown, NJ, USA, 2008. Association for Computational Linguistics.

[82] X. Sun and J. Tsujii. Sequential labeling with latent variables: an exact inference algorithm and its efficient approximation. In *EACL '09: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 772–780, Morristown, NJ, USA, 2009. Association for Computational Linguistics.

[83] C. Sutton and A. McCallum. An Introduction to Conditional Random Fields for Relational Learning. In *Introduction to Statistical Relational Learning*. MIT Press, Cambridge, MA, USA, 2006. Last retrieved on October 13, 2010 from `http://www.cs.umass.edu/~mccallum/papers/crf-tutorial.pdf`.

[84] K. Swaminathan and S. Sato. Interaction design for large displays. *ACM Interactions magazine*, 4(1):15–24, 1997.

[85] M. E. Taylor and P. Stone. Transfer Learning for Reinforcement Learning Domains: A Survey. *The Journal of Machine Learning Research*, 10:1633–1685, December 2009.

[86] M. Turk. *Handbook of Virtual Environment Technology*, chapter 10. Lawrence Erlbaum Associates, Inc., 2001. Last retrieved on July 15, 2010 from http://ilab.cs.ucsb.edu/projects/turk/TurkVEChapter.pdf.

[87] University of Las Palmas. *GpdsHMM Toolbox User's Guide*, 2010. Last retrieved on October 2, 2010 from http://www.gpds.ulpgc.es/download/toolbox/gpdshmm/gpdsHMM%20toolbox%20for%20matlab%20v1.0.pdf.

[88] A. van Dam. Post-WIMP user interfaces. *Communications of the ACM*, 40(2):63–67, 1997.

[89] M. van Doorn, E. van Loenen, and A. P. de Vries. Deconstructing ambient intelligence into ambient narratives: the intelligent shop window. In *Ambi-Sys '08: Proceedings of the 1st international conference on Ambient media and systems*, pages 1–8, ICST, Brussels, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[90] D. Vogel and R. Balakrishnan. Distant freehand pointing and clicking on very large, high resolution displays. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 33–42, New York, NY, USA, 2005. ACM.

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

[92] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers, San Francisco, CA, USA, 2005.

[93] J. Wu, G. Pan, D. Zhang, G. Qi, and S. Li. Gesture Recognition with a 3-D Accelerometer. In *UIC '09: Proceedings of the 6th International Conference on Ubiquitous Intelligence and Computing*, pages 25–38, Berlin, Germany, 2009. Springer Verlag.

[94] Y. Wu and T. S. Huang. Vision-Based Gesture Recognition: A Review. In *GW '99: Proceedings of the International Gesture Workshop on Gesture-Based Communication in Human-Computer Interaction*, pages 103–115, London, UK, 1999. Springer Verlag.

[95] L. Xi, Y. Fondufe-Mittendorf, L. Xia, J. Flatow, J. Widom, and J.-P. Wang. Predicting nucleosome positioning using a duration Hidden Markov Model. *BMC Bioinformatics*, 11(1):346–354, 2010.

[96] J. Yamato, J. Ohya, and K. Ishii. Recognizing human action in time-sequential images using hidden Markov model. In *CVPR '92: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 379–385, 15-18 1992.

[97] J. Yang and X. Yangsheng. Hidden Markov Model for Gesture Recognition. Technical Report CMU-RI-TR-94-10, Robotics Institute, Carnegie Mellon Uni-

versity, Pittsburgh, PA, USA, May 1994. Last retrieved on May 3, 2010 from http://www.ri.cmu.edu/publication_view.html?pub_id=329.

[98] H.-S. Yoon, J. Soh, Y. J. Bae, and H. S. Yang. Hand gesture recognition using combined features of location, angle and velocity. *Pattern Recognition*, 34(7):1491–1501, 2001.

[99] T. G. Zimmerman, J. Lanier, C. Blanchard, S. Bryson, and Y. Harvill. A hand gesture interface device. *SIGCHI Bulletin*, 17(SI):189–192, 1987.

[100] M. Zobl, R. Nieschulz, M. Geiger, M. Lang, and G. Rigoll. Gesture Components for Natural Interaction with In-Car Devices. In *Gesture-Based Communication in Human-Computer Interaction: 5th International Gesture Workshop*, volume 2915 of *Lecture Notes in Computer Science*, pages 367–368. Springer Verlag, Heidelberg, Germany, 2004.

Appendices

Appendix A

Flock Of Birds Quick Manual

The aim of this manual is to briefly explain the necessary configurations to read position and orientation data from the Ascension Flock of Birds system[1] (FOB) in the Java programming language. It was written as an appendix to a master thesis [2] Figure A.1 shows the FOB system this manual applies to.

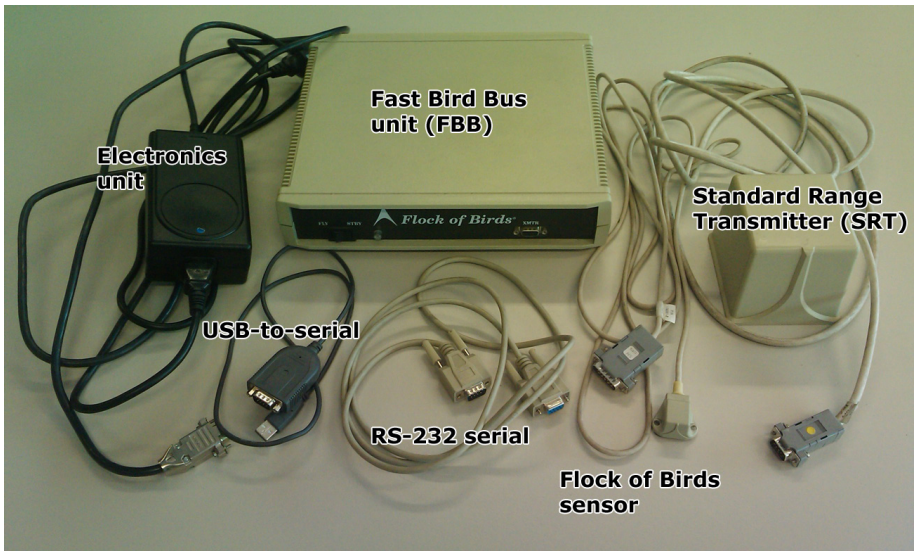


Figure A.1.: *The Ascension Flock of Birds system used.*

All these components (except for the USB-to-serial perhaps) should be available to you. You might also have purchased additional Fast Bird Buses (FBBs) and

[1] See http://www.ascension-tech.com. Last visited on January 6, 2010.
[2] Thesis 'User-independent recognition of dynamic hand gestures using a dataglove' by M. Ganzeboom, October, 2010.

FOB sensors which you would like to use. This manual starts with explaining how to configure the hardware in multiple contexts and continues with giving instructions on how to read from the FOB system through the Java driver. That driver should have been bundled with this manual. Throughout this manual, references are made to the official FOB installation and operation guide revision C (hereafter I&O guide), which is supplied to you with this manual or can be downloaded from Ascension Tech's FTP servers[3].

A.1. Configuring the FOB hardware

Assuming that multiple FBBs and FOB sensors are at your disposal, the FOB system can be configured in two ways depending on the context the FOB system is going to be used. If your context asks for the use of multiple FOB sensors, the system needs to be configured in 'FOB mode'. If only a single FOB sensor is required this manual recommends 'Standalone mode'. This manual was written as part of a master thesis on hand gesture recognition in which the FOB system was used in standalone mode. For that reason, the main focus is on configuring for that mode. Instructions are derived for FOB mode, but they are not based on practical experiences.

Standalone mode
Collect all the components shown in Figure A.1. Take the FBB and connect the Standard Range Transmitter (SRT) to the front port. Connect the FOB sensor, RS-232 serial cable and the Electronics Unit (EU) to the right ports on the back of the FBB.

FOB mode
Connect the FOB sensors and EUs to all FBBs as described under standalone mode. Interconnect the FBBs with the FBB inter-unit bus cables (Figure 2, I&O guide, pg. 4) and connect the SRT to the master FBB (the first FBB).

Connect the first (or only) FBB to a PC. Do not yet plug in the power cord of the EU to the wall socket. First, take a look at the eight dipswitches on the backpanel of the FBB (switches under the label 'DIP SWITCH'). To be in normal operating mode (and not in test mode), dipswitch 8 must be in the up position on all FBBs to be used (I&O guide 2.1.7, pg. 10). We assume that the 'Normal Addressing Mode' for giving each Bird unit (i.e. SRT or FOB sensors) a unique

[3] See ftp://ftp.ascension-tech.com/MANUALS/Flock_of_Birds_Manual-RevC.pdf. Last downloaded on October 14, 2010.

address is sufficient for your context (supports up to one SRT and 13 FOB sensors). For other addressing modes see Subsection 2.1.7 of the I&O guide, page 11.
To use the maximum FOB baud rate (115200 baud), dipswitches 1, 2 and 3 should be in the down position. Note that all FBBs in the setup must use the same baud rate. For setting the FOB baud rate to other settings see Figure 4 on page 12 of the I&O guide. Dipswitches 4, 5, 6 and 7 are used to define the address of a Bird unit connected to the FBB.

Standalone mode
In this mode one FBB, FOB sensor, SRT and corresponding cables are to be used. Dipswitches 4, 5, 6 and 7 must all be in the up position, which gives the FOB sensor the address 0.

FOB mode
Multiple FBBs and FOB sensors are to be used in this mode with a single SRT, assuming that the FBBs are interconnected as described above. Configure dipswitches 4, 5, 6 and 7 on each FBB so that each connected FOB sensor has a different address. Which address is assigned to a FBB is not important, as long as the assigned addresses are continuous (i.e. no address is skipped). The addresses must also be set to 1 or higher (else single mode operation is configured). For example, to set the address of a FOB sensor to 1 flip the dipswitches 4, 5, 6 and 7 to up, up, up, down. The second sensor then has its address set to 2 by flipping the same switches to up, up, down, up. The third is set to 3 by up, up, down, down and so on.

On the front of every FBB there is a switch which can be set to either 'FLY' or 'STBY'. This switch is used to let the FBB perform its power up functions. It is not an on/off power switch. To cut the power to the FBBs unplug the power cord. Flip the switch to 'STBY' if not in that position already. Then plug the power cord of all EUs into sockets.

Standalone mode
Flip the switch of the single FBB to the 'FLY' position. The (red) light indicator next to the switch will blink on and off 5 times and then stays on. At that time, the SRT and sensor will start operating and the FBB is ready to accept commands from the host computer. When the switch is flipped to 'STBY' the light indicator turns off, the SRT and sensors are shut off and the FBB wil not respond to any host computer commands.

FOB mode

Flip the switches of all FBBs one-by-one to the 'FLY' position (the order does not seem to be important) and check that the (red) light indicator next to the switch will blink on and off 5 times and then goes off. The host computer must then send the master FBB an AUTO-CONFIG command (explained later). On receipt of this command the light indicators on all FBBs will turn on and stay on continuously without blinking if operating correctly. If indicators do blink, refer to Section 11.0 of the I&O guide for the error codes.

For more details on the above power up process refer to Subsection 5.2.1 of the I&O guide. Assuming that the hardware is powered up properly, the configuration of the hardware is now complete. The next section explains how to use the driver.

A.2. Accessing the FOB

Accessing the Flock Of Birds (FOB) system through the Java programming language is done with the driver supplied with this manual. The driver consists of a single Java class in one file called 'FOBDriver.java'. With this driver the Bird units (i.e. FOB sensors and SRT) can be controlled through the Fast Bird Bus (FBB) from the host computer. The commonly used control commands described in the I&O guide are implemented. This section explains how a connection to the FOB system is established from the host computer, how the FOB software is properly configured and how position and orientation data can be read from the FOB. We assume that the reader is experienced in programming in Java and therefore omit irrelevant details about the Java syntax and setup. From this point, the FOB hardware is assumed to be correctly configured in one of the two configurations as described in Section A.1.

Establishing a connection with the FOB hardware

Communicating with the FOB system is done via the RS-232 serial protocol. To be able to access the serial ports from Java the RXTX library is needed. Download the binaries from the RXTX website [4]. At time of writing, release 2.1-7r2 is the most recent stable version and was used in writing this manual. Put the binary files for your specific operating system on the java library path (tip: java property java.library.path) and the 'RXTXcomm.jar' on the class path (tip: environment variable 'CLASSPATH'). Establishing a connection between the FOB hardware and the host computer from Java is done in the following way:

[4] See http://rxtx.qbang.org/wiki/index.php/Download. Last visited on October 14, 2010.

1. Integrate the `FOBDriver.java` file into your Java project.
2. Instantiate a new `FOBDriver` object via the default (empty) constructor.
3. Determine the serial port of the host computer to which the FOB system is connected.
4. At the time you want to establish a connection, call the `FOBDriver.open(String port)` method with the serial port number from step 3 as parameter.
5. Pause the Java thread that establishes the connection for about 2000 milliseconds. This is necessary to apparently give the FOB system enough time to initialise properly.
6. See the API comments of the open method to interpret the status value returned.

After the connection is successfully established, the FOB system needs to be configured to return the form of position and orientation information you require. The 3D position information is read by the FOBDriver in an x, y, z coordinate and the orientation in either axis angles, a matrix or a quaternion (see Section 9.0 of I&O guide). Currently, the FOBDriver supports to set the output to the combinations POSITION/ANGLES and POSITION/QUATERNION. However, methods for the other modes are easily implemented. Depending on the FOB system being setup in standalone or FOB mode the following instructions apply:

Standalone mode

1. To let the FOB system output position and orientation information in the format of POSITION/QUATERNION, call the `FOBDriver.setSingleFlockToPosQuat()` method after having successfully established a connection. Use the `FOBDriver.setSingleFlockToPosAngle()` to set the output to POSITION/ANGLE format.
2. Depending on the placement of the SRT in your total system setup, different hemispheres of the magnetic field the SRT outputs can be enabled. In our example the SRT is placed on the floor so only the upper hemisphere is required. To configure that call `FOBDriver.setHemisphereForSingleFlock(int hem)` method with the parameter `FOBDriver.UPPER_HEMISPHERE`. (Parameters defining other hemispheres are similarly defined in `FOBDriver.java`).
3. To expand the range of the FOB sensors from the default 36 inch to 72 inch, the `FOBDriver.expandRangeCommand(int nrOfBirds)` method

should be called with the number of FOB sensors in use as the parameter.

4. To request the current position and orientation in the format just configured, call the method `FOBDriver.getPointInStandardGraphicMode(Tuple3f pos, Quat4f orient)`. The returned position and orientation are already converted from the Ascension to the standard computer graphics axes alignment. (see API comments and I&O guide, pg. 152)

FOB mode

1. Call the method `FOBDriver.sendMasterFBBAutoConfigCommand(int nrOfBirds)` to send the master FBB the AUTO-CONFIG command. Indicator lights on the front panel of all FBBs turns on and stays on continuously.
2. To let the FOB system output position and orientation information in the format of POSITION/QUATERNION, call the `FOBDriver.setMultiFlock(int nrOfBirds)` method after having successfully established a connection and use the total number of birds in use as the parameter. No method has been implemented to let the FOB system output other formats.
3. Depending on the placement of the SRT in your total system setup, different hemispheres of the magnetic field the SRT outputs can be enabled. In our example the SRT is placed on the floor so only the upper hemisphere is required. To configure that call `FOBDriver.setHemisphereForMultiFlock(int hem, int nrOfBirds)` method with the parameter `FOBDriver.UPPER_HEMISPHERE` and the total number of birds in use as the second parameter.
4. To expand the range of the FOB sensors from the default 36 inch to 72 inch, the `FOBDriver.expandRangeCommand(int nrOfBirds)` method should be called with the number of FOB sensors in use as the parameter.
5. Send the master FBB the AUTO-CONFIG command, like in step 1.
6. To request the current position and orientation of a specific FOB sensor, call the method `FOBDriver.getPoint(int birdNr, Tuple3f pos, Quat4f orient)`. The returned position and orientation are not yet converted to the standard computer graphics axes alignment (see API comments and I&O guide, pg. 152).

7. Use the `FOBDriver.convertToStandardGraphicMode(Vector3f ascVector)` and `FOBDriver.convertToStandardGraphicMode(Quat4f ascQuat)` methods to convert them into the standard computer graphics axes alignment when required.

That finishes this section on accessing the FOB system with the necessary commands. For more command reference see Section 9.0 and possible errors (e.g. continuous blinking of FBB indicator lights) in Section 11.0 of the I&O guide.

Appendix B

Data recording protocol

After having greeted and instructed the participant of the data recording experiment, the dataglove and Flock of Birds sensor were put on. The participant practiced a few minutes, to get a feeling for the cursor control. When the participant was ready, the researcher started the recording process and followed the protocol below.

1. Press the 'Y' key once on the keyboard to enable cursor control to the participant
2. While controlling the CoolIris application and there are less than fifteen instances of every gesture, do:
   a) If gesture Select is observed, press the 'W' key once to execute a left mouse button click
   b) Else if the start of gesture Drag is observed, press the 'W' key once to execute a left mouse button press
   c) Else if the end of gesture Drag is observed, press the 'W' key once to execute a left mouse button release
   d) Else if the Zoom in gesture is observed, press the 'E' key repeatedly until the participant stops the gesture or pauses moving thumb and index finger
   e) Else if the Zoom out gesture is observed, press the 'D' key repeatedly until the participant stops the gesture or pauses moving thumb and index finger
   f) Else if the Maximise gesture is observed, press the 'O' key once
   g) Else if the Demaximise gesture is observed, press the 'D' key once
   h) Else if the participant makes an undefined gesture or one that devi-

ates from its description, say out loud that the gesture is not recognised

3. Press the 'Y' key once on the keyboard to disable cursor control to the participant
4. Stop the recording and change onscreen application to the Movie Memory application
5. Ask the participant to centre the Flock of Birds sensor above the magnetic field generator
6. Start the recording and press the 'Y' key once on the keyboard to enable cursor control to the participant
7. While controlling the Movie Memory application and there are less than fifteen instances of every gesture, do:
   a) If gesture Select is observed, press the 'W' key once to execute a left mouse button click
   b) Else if the Minimise gesture is observed, press the 'P' key once
   c) Else if the Deminimise gesture is observed, press the 'L' key once
   d) Else if the Close gesture is observed, press the 'S' key once
   e) Else if the participant makes an undefined gesture or one that deviates from its description, say out loud that the gesture is not recognised
8. Press the 'Y' key once on the keyboard to disable cursor control to the participant
9. Stop the recording

Appendix C

Characteristics of experiment participants

To train a hand gesture recogniser an experiment was held to collect data. The experiment is explained in Chapter 3. Data was recorded from 22 participants. The target group was students and young academic researchers. The table below shows their characteristics and previous experience with gesture interfaces. The rows marked bold and italic are the participants from which the data was used in training and testing the recognisers in Sections 5.1 to 5.4. The data of participants eleven and fifteen was used in Section 5.5 for testing the user independence.

#	Age	Gender	Nation.	Occupation	Left-handed	Experience
1	20-30	M	Dutch	Ph.D. student	No	Gesturing with Wii and multi-touch phones
2	20-30	M	Dutch	MSc. student	No	Gesturing with multi-touch screens
3	***20-30***	***M***	***Dutch***	***MSc. student***	***No***	***Gesturing with multi-touch phones***
4	***20-30***	***M***	***Dutch***	***MSc. student***	***No***	***Gesturing with Wii and multi-touch phones***
5	20-30	M	Iranian	Bachelor student	No	None

#	Age	Gender	Nation.	Occupation	Left-handed	Experience
6	10-20	F	Dutch	Bachelor student	No	Gesturing with multi-touch phones
7	20-30	M	Dutch	MSc. student	No	Gesturing with multi-touch phones, multi-touch table
8	20-30	M	Dutch	MSc. student	Yes	Gesturing with multi-touch table and screens
9	***20-30***	***M***	***Dutch***	***Postdoc researcher***	***No***	***None***
10	10-20	M	Dutch	BSc. student	No	None
11	***20-30***	***M***	***German***	***Bachelor student***	***No***	***Mouse gestures, gesturing with multi-touch screens***
12	20-30	M	Dutch	MSc. student	No	Gesturing with multi-touch phones and screens
13	20-30	M	Dutch	MSc. student	No	Gesturing with multi-touch phones, touch screens and mid-air vision-based gesture recognition
14	20-30	M	Dutch	MSc. student	Yes	Gesturing with multi-touch phones
15	***20-30***	***M***	***Dutch***	***MSc. student***	***No***	***None***
16	20-30	M	German	BA student	No	None
17	30-40	M	Dutch	Ph. D. student	Yes	Gesturing with multi-touch phones
18	***30-40***	***M***	***Dutch***	***Ph. D. student***	***No***	***Gesturing with Wii, multi-touch screens and mid-air, bimanual laser-pointer gesture recognition***

#	Age	Gender	Nation.	Occupation	Left-handed	Experience
19	***20-30***	***M***	***Dutch***	***MSc. student***	***No***	***Gesturing with multi-touch phones and screens***
20	20-30	M	Dutch	Ph. D. student	No	Gesturing with Wii and 3D Mouse
21	***30-40***	***M***	***Dutch***	***Postdoc researcher***	***No***	***Gesturing with Wii and bimanual laser-pointer gesture recognition***
22	20-30	M	Dutch	MSc. student	No	Gesturing with multi-touch screens and mid-air vision-based gesture recognition