Die approbierte Originalversion dieser Diplom-/Masterarbeit ist an der Hauptbibliothek der Technischen Universität Wien aufgestellt (http://www.ub.tuwien.ac.at).

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology (http://www.ub.tuwien.ac.at/englweb/).

FAKULTÄT
FÜR !NFORMATIK

Faculty of Informatics

Hand/Arm Gesture Recognition based on Address-Event-Representation Data

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Medizinische Informatik

eingereicht von

Matthias Zima
Matrikelnummer 0227386

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Robert Sablatnig
Mitwirkung: DI Dr. Martin Litzenberger

Wien, 08.10.2012

(Unterschrift Verfasserin)

(Unterschrift Betreuung)

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.ac.at

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Hand/Arm Gesture Recognition based on Address-Event-Representation Data

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Medical Informatics

by

Matthias Zima
Registration Number 0227386

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Robert Sablatnig
Assistance: DI Dr. Martin Litzenberger

Vienna, 08.10.2012

(Signature of Author)

(Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Matthias Zima
Hannovergasse 19/32, 1200 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum)	(Unterschrift Verfasserin)

Danksagung

Mit dieser Arbeit geht einer der längsten Abschnitte meines Lebens zu Ende. Ich möchte dieser Gelegenheit nutzen um mich bei all jenen zu bedanken, die mir in dieser Zeit zur Seite standen. Sollte ich jemanden vergessen bitte ich das zu verzeihen, da mich in dieser Zeit sehr viele Menschen begleitet haben. Als erstes möchte ich meinen Eltern und meiner Familie danken, die auch in schwierigen Zeiten immer hinter mir standen und mich in meinem Weg unterstützten. Außerdem möchte ich mich bei Eva-Maria Hanzl bedanken, die mir die Kraft und die Inspiration gab das Studium auch neben der Berufstätigkeit zu einem Abschluss zu bringen. Hinsichtlich der Berufstätigkeit möchte ich mich bei meinem Arbeitgeber, der Firma Kapsch TrafficCom AG, bedanken, welche es mir durch Rücksichtnahme auf meine Bedürfnisse ermöglichte diese Arbeit zu vollenden. Ein großer Dank richtet sich an das Austrian Institute of Technology , welches mir die Durchführung dieser Arbeit im Rahmen des Projektes FoSIBLE ermöglichte. Dank gebührt auch Dr. Martin Litzenberger vom Austrian Institute of Technology für die Betreuung und die Geduld, die er mit mir hatte. Zu guter Letzt möchte ich auch der TU Wien für die interessante und inspirierende Studienzeit danken.

Abstract

With the appearance of Microsofts Kinect sensor gesture recognition has become a commonly discussed topic. Since gestures are a natural form of communication used by humans, interfaces that are controlled by gestures recognition is a promising way to improve the interaction between humans and computers. Therefore also the FoSIBLE[1] project, part of the AAL[2] program, introduces a gesture controlled user interface to ease up the interaction for elderly people using a digital system. But not only new user interfaces are interesting in context with gesture control, but also the question how the motion of humans is captured has different answers.

In this thesis it is evaluated, if the AE[3] data generated by the UCOS[4] sensor can be used for hand/arm gesture recognition. The results from this are a first step for the future development of an embedded gesture recognition device based on the UCOS sensor. The UCOS sensor is a novel biologically inspired 3D sensing device developed by AIT[5] that senses scene dynamics and exclusively transmits edge information of moving objects while hiding static areas. The so gathered pixel information data are communicated via a sparse, asynchronous protocol called address-event-representation. Trajectories of moving objects like hands/arms are also encoded within this address-event data. A specially modified firmware for gesture recognition enables filtering these trajectories out of the address-event stream. Based on this data features are calculated which are used to classify the gestures using different machine learning methods, the Hidden Markov Model and the Decision Tree. Both classification methods are trained with different sized sets of gestures containing ten, eight and four gestures. From each gesture at least 52 samples are used to evaluate the gesture recognition. The results of the Hidden Markov Model and the Decision Tree are compared with each other.

For Evaluation ten different gestures (eight directional gestures, a select gesture and a return gesture) which are intended to control a multimedia interface were defined. Totally 1463 gestures were recorded, annotated and stored in a gesture database. The classifiers were trained with gestures from this database. Varying the parameters and using cross validation following results were obtained for both methods: Recognition rate with ten gestures approx. 90%, recognition rate with eight gestures approx. 95% and recognition rate with four gestures approx. 100%.

---

[1] Fostering Social Interaction for the Well-Being of the Elderly

[2] Ambient Assisted Living

[3] Adress Event

[4] Universal COunting Sensor

[5] Austrian Institute of Technology

Kurzfassung

Gestenerkennung ist spätestens seit dem Erscheinen vom Microsofts Kinect Sensor ein weitreichend bekanntes Thema. Da Gesten ein vom Menschen natürlich eingesetztes Kommunikationsmittel darstellen verspricht man sich von auf Gestenerkennung basierenden Benutzerinterfaces eine Verbesserung der Interaktion zwischen Menschen und Computern. Auch im Zuge des AAL[6] Projektes FoSIBLE[7] soll eine gestenbasierte Benutzeroberfläche älteren Menschen die Interaktion mit einem digitalen System erleichtern. Aber nicht nur neue Benutzerinterfaces sind von Interesse, auch die Frage wie die Bewegung von Menschen erfasst wird kann auf vielfältige Art und Weise gelöst werden.

In dieser Diplomarbeit wird untersucht, ob die vom UCOS[8] Sensor generierten AE[9] Daten zur Erkennung von Hand/Arm Gesten verwendet werden können. Die Arbeit dient als Grundlage für das zukünftige Ziel, auf Basis des UCOS Sensors ein integriertes System zur Gestenerkennung zu entwickeln. Der UCOS Sensor ist ein neuartiger, vom AIT[10] entwickelter, biologisch inspirierter 3D Sensor, welcher auf temporalen Kontrast aufgrund von Szenendynamiken reagiert und die so gewonnenen Pixelinformationen auf in Form von Adress-Events auf asynchroner Basis sendet. Innerhalb dieser Adress-Events sind auch Trajektorien von bewegten Objekten, wie beispielsweise Händen, encodiert. Eine speziell für diese Anwendung modifizierte Firmware ermöglicht es, diese Trajektorien aus dem Adress-Event Strom zu filtern. Aus diesen Daten wurden grundliegende Features berechnet, welche dann für eine Klassifikation durch Hidden Markov Modelle und durch einen Entscheidungsbaum verwednet wurden. Die beiden Klassifikationsmethoden wurden mit verschieden großen Gesten-Sets bestehend aus zehn, acht und vier vordefinierten Gesten trainiert. Jedes der Gesten-Sets enthielt mindestens 52 Samples. Die Ergebnisse beider Methoden wurden miteinander verglichen.

Für die Evaluierung wurden zehn verschiedene Gesten (acht direktionale Gesten, eine Auswahlgeste und eine Retourgeste), welche zur Steuerung eines Multimediainterfaces vorgesehen sind, definiert. Bei Testaufnahmen wurden 1463 Gesten aufgenommen und nach einer manuellen Annotation in einer Gestendatenbank gespeichert. Diese Daten wurden dann zum Trainieren der Klassifikatoren verwendet. Mittels verschiedener Parameter wurde dann eine Kreuzvalidierung

---

[6] Ambient Assisted Living

[7] Fostering Social Interaction for the Well-Being of the Elderly

[8] Universal COunting Sensor

[9] Adress Event

[10] Austrian Institute of Technology

der Algorithmen durchgeführt. Dabei ergaben sich mit beiden Methoden für zehn Gesten Erkennungsraten um die 90%, für acht Gesten Erkennungsraten um 95% und für vier Gesten Erkennungsraten nahe 100%.

Contents

1 Introduction 1
1.1 Motivation 1
1.2 Problem Statement and Scope of Work 3
1.3 Methodological Approach 4
1.4 Main Contribution 4
1.5 Thesis Structure 5

2 Related Work and State of the Art 7
2.1 Human Computer Interaction 8
2.2 Gestures 10
2.3 Gesture Recognition 13
2.4 Works on Gesture and Motion Recognition 22
2.5 Summary of Related Work and State of the Art 32

3 Methodology 33
3.1 Dynamic Stereo Vision Sensor for Gesture Recognition 34
3.2 Simple GUI Concept for Gesture Recognition 39
3.3 Defintion of Specific Gestures 42
3.4 Data Recording and Preprocessing 45
3.5 Test Database and Data Annotation 49
3.6 Features 51
3.7 Machine Learning Based Gesture Recognition 57
3.8 Cross Validation 65
3.9 Summary of Methodology 66

4 Implementation 67
4.1 Implementation of the HMM classification 67
4.2 Implementation of the DT (Quasi Rule Based) Classification 70
4.3 Feature Calculation 71
4.4 Summary of Implementation 72

5 Experimental Results 73
5.1 Visual Evaluation of Features 73

5.2 Preface for Classification . . . 76
5.3 Classification with the Hidden Markov Models . . . 77
5.4 Classification with the Decision Tree (Quasi Rule Based) . . . 84
5.5 Summary of Results and Comparison between HMM and DT . . . 92

6 Conclusion and Future Work 93
6.1 Conclusion and Discussion of the Results . . . 93
6.2 Future Work . . . 94

A Appendix A: Feature Plots 97

B Appendix B: Code 105
B.1 Appendix B, Code: Filter different AER types . . . 105
B.2 Appendix B, Code: HMM Chain . . . 106
B.3 Appendix B, Code: DT Chain . . . 110

C Appendix C: Sensor Settings 115
C.1 Settings of the UCOS Sensor . . . 115

Bibliography 119

Acronyms in Alphabetical Order

AAL	Ambient Assisted Living	1
ADDR	Address	37
AE	Adress Event	3
AER	Address Event Representation	1
AIT	Austrian Institute of Technology	1
ATIS	Asynchronous Time based Image Sensor	22
BB	Bounding Box	54
CMOS	Complementary Metal Oxide Semiconductor	35
CPU	Central Processing Unit	35
DCT	Discrete Cosine Transformation	69
DT	Decision Tree	4
FCA	Feature Cell Array	
FoSIBLE	Fostering Social Interaction for the Well-Being of the Elderly	1
GUI	Graphical User Interface	5
HCI	Human Computer Interaction	8
HMM	Hidden Markov Model	4
HRI	Human Robot Interaction	14
ID	Identifier that uniquely identifies an object or record	49
LAN	Local Area Network	35
MATLAB	Matrix Laboratory, a numerical computing environment and fourth-generation programming language developed by MathWorks	4
MLP	MuliLayer Perceptron, an artificial neural network	18
NN	Neuronal Networks	14
PC	Personal Computer	14
RAM	Random Access Memory	35
SAD	Sum of absolute differences	35

SVM	Support Vector Machines	4
TAE	Timed Address Event	35
TS	Time Stamp	37
TV	Television	39
UCOS	Universal COunting Sensor	1
UDP	User Datagram Protocol	35
3D	Three-Dimensional space	14

CHAPTER 1

Introduction

The target of this work is to evaluate, if the AER[1] data generated by AIT[2]s UCOS[3] sensor can be used for gesture recognition. This evaluation is done as part of the AAL[4] [1] project FoSIBLE[5], because the development of an alternative natural and easy-to-use input method for a multimedia interface is one of the aims of the project [4]. FoSIBLE was established to support and relieve social interaction in the daily life of the elderly with regard to their individual living circumstances, needs and interests. According to the research conducted for FoSIBLE, many people prefer aging at home over institutional care facilities [52], whereas remaining in the community in later life can be problematic - not only because of medical problems, but also because of the loss of companionship [52]. The loss of companionship can lead to isolation, depression, decreased socialization and may have negative impacts on the general health status [52]. While social support aims to facilitate interaction within the neighborhood, social interactions beyond the near environment with remotely living family members and friends also seems to be important [4].

1.1 Motivation

Gestures are an integral part of human communication [39]. They are (wittingly or unwittingly) used to send signals to other humans and the environment without using words and without making sounds. Be it for enhancement or simple support of human communication, gesturing is natural in human behavior [24] [39]. Therefore, gesture recognition promises to provide an easy-to-use, intuitive and natural interface for the interaction with computers or electronic systems in general [62]. Especially the arrival of Microsoft's Kinect 3D-sensing device on the

---

[1] Address Event Representation

[2] Austrian Institute of Technology

[3] Universal COunting Sensor

[4] Ambient Assisted Living

[5] Fostering Social Interaction for the Well-Being of the Elderly

consumer market has triggered a boom in the development and demonstration of gesture control interfaces [14]. In 2012 at least 350 companies are developing custom applications that are using the gesture recognition capabilities of Microsoft's Kinect [14]. With the increased interest in gesture recognition it is also interesting to evaluate another technological approach for sensing human motion in this field of application. Therefore the aim of this work is to evaluate whether it is possible to use AIT's UCOS sensor for gesture recognition. A comparison of the technological sensor approaches is not part of this work.

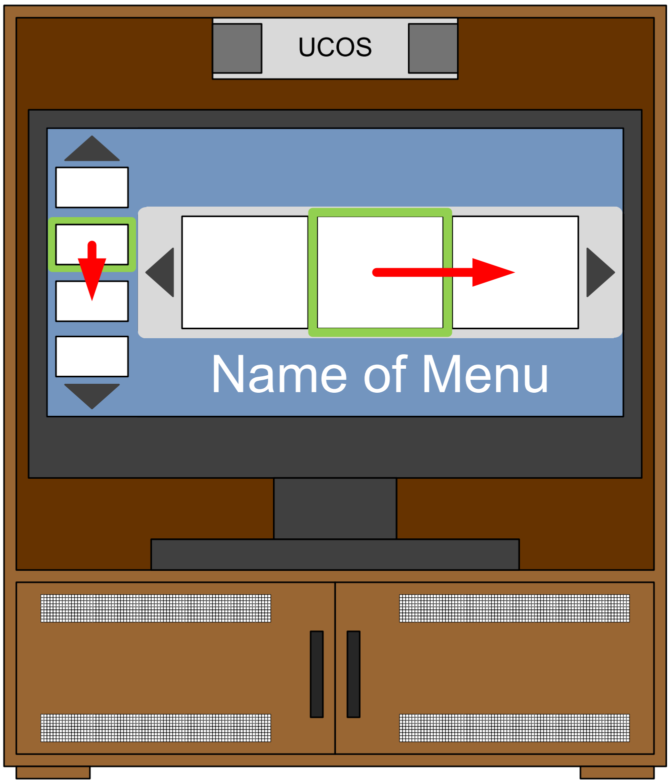


Figure 1.1: Early concept of intelligent furniture with an integrated console-based community platform controlled by gestures

The development of a console-based community platform and focusing on sensor-based haptic input and output devices and intelligent furniture in order to enable shared experiences with others is a major task of the FoSIBLE project [4]. An example for early concepts of a intelligent piece of furniture is given in Figure 1.1, where a TV-set and the sensor for gesture control is mounted together in a cupboard. The aim to integrate gestures for controlling the FoSIBLE application is to offer an alternative to the main input method based on a tablet device. The users should not need any additional device, they can immediately start to interact with the system by just moving their hands. Searching the remote control devices will be a thing of the past, since the basic controls will be available without additional control devices. Gestures are a natural input modality that evolved from real-world interaction styles being more intuitive and easier to learn than indirect input modalities like remote controls [38]. According to a study which evaluated performance and acceptance of older adults using freehand gestures for TV Menu control [20] the persons participating had a very positive attitude towards gesture-based interactions. This study supports the conclusion, that gesture recognition is a promising input method for the elderly. Under this circumstances the verification of a novel sensor device for a new application - gesture recognition - also becomes a topic worth to take a further look on.

1.2 Problem Statement and Scope of Work

AIT has developed biomimetic inspired stereo vision sensors, which use the AER data format. Information on local intensity changes is detected and measured asynchronously and individually by each pixel at temporal resolution in the microsecond range. The operation principle results in highly efficient compression of visual data through temporal redundancy suppression at the focal-plane [59]. Data are communicated via a sparse, asynchronous, event based protocol, the address-event-representation, or AER. By its nature, this vision-sensor technology is especially suited for applications involving accurate detection of continuous motion like the tracking of hand/arm movement in gestures. The AE[6] representation is fundamentally different from the image frame data that are usually produced by conventional digital systems at a fixed frame rate [13].

Currently the biomimetic stereo vision sensor is used for counting persons, therefore it is named Universal COunting Sensor (UCOS). Gesture recognition is a promising new application for this stereo vision sensor. The target of this master thesis is to develop computation methods for efficient processing of AER data for the recognition of hand/arm gestures. This means that software algorithms for the extraction of features and for gestures recognition on the AER data stream have to be developed.

The outcome of this work evaluates if the AER-Data produced by the UCOS sensor can be used for dynamic hand/arm gesture recognition. Also two different training and recognition concepts will be compared: a machine-learning and a rule-based approach. The rule-based approach has the advantage, that (a possible future embedded) implementation of the gesture recognition algorithms in the device will not take too much effort, since the rules can easily be transferred into the embedded environment. A machine learning approach is more state-of-the-art and it also led to promising results in preceding works [42] [67]. A system ready to be released on the consumer market is not the aim of this work.

---

[6] Adress Event

1.3 Methodological Approach

The investigation is based on the AER data produced by the existing UCOS stereo sensor system, which is actually designed to count people passing by underneath it. The work covers the whole process from data acquisition to gesture recognition. A specific set of simple hand/arm gestures is used for this study. AIT has already implemented some basic algorithms to process AER data. This algorithms that are implemented in Mathworks MATLAB[7]. As basic format conversion functions of those algorithms are used and/or extended during this work, Mathworks MATLAB is used for all calculations and programming tasks in this thesis.

Tracking the closest object that moves next to the sensor is already implemented in the embedded firmware of the device. The extracted tracks are also encoded in the UCOS sensor's AER data stream. When moving a hand/arm in front of the device (performing a hand/arm gesture) the sensor will continuously output the position of the hand/arm given that it is the closest object to the sensor.

The resulting tracks have to be cleaned, filtered and preprocessed. The data from the preprocessed tracks is used to extract features that may contain velocity, direction and shape of the movements. Then it will be analyzed which of the features are useful for gesture recognition. It is expected that combinations of features - for example direction an shape of the movement - resemble individual gestures, which should be detected by the gesture recognition algorithms.

For all test and evaluation activities test data was recorded, evaluated and categorized. A database for the track data had to be created. In the first attempt a rule-based approach was used to evaluate the features and to detect the gestures. Furthermore the features will be evaluated using a stochastic machine learning model (HMM[8], SVM[9], ...) to detect the different gestures that were performed. The results of the-rule based and the machine-learning approach are evaluated individually and are then compared against each other.

1.4 Main Contribution

The evaluation, if the AER data generated by the UCOS sensor can be used for gesture recognition, represents the main contribution of this work. The focus is, how this data can be used,to automatically train machine-learning based classifiers for gesture recognition. Two different machine learning approaches are used to demonstrate gesture recognition using the UCOS sensor, HMM and DT[10]. The HMM is chosen, because it has the ability to model the time aspect of the features describing a gestural hand/arm motion. The DT was chosen, because it automatically generates a set of rules, that can be transferred to other programming environments (i.e.

---

[7] Matrix Laboratory, a numerical computing environment and fourth-generation programming language developed by MathWorks

[8] Hidden Markov Model

[9] Support Vector Machines

[10] Decision Tree

C or others used for embedded implementation) with little effort, because this rules consist of widely-used “if-then” statements.

The main contribution does not focus on the technological characteristics of the UCOS sensor, but on the processing of the AER data and the algorithms for machine-learning based gesture recognition. One function of the UCOS sensor is to track hands/arms that are moving in front of the sensor. Trajectories of this movements are encoded within the AER data generated by the sensor. This trajectories have to be filtered out of the AER stream to be used for gesture recognition, which was implemented within this work.

For this thesis a tool to annotate the recordings that are used for training an evaluation of the classifiers, had to be implemented. Another contribution is the selection of features for gesture recognition. The methods to calculate the features, a visual inspection of features and a discussion on their usability for gesture recognition were made for this thesis. Due to the characteristics of the UCOS sensor, only dynamic gestures can be recognized [13]. Taking this into account a GUI[11]-concept and gestures based on this GUI-concept (eight directional gestures, a select gesture and a return gesture) were defined to be used for training and evaluation. Gestures were recorded and annotated resulting in a gesture database containing 1463 samples. This samples were used to train the two machine-learning methods, HMM and DT which are proposed for gesture recognition in this thesis.

1.5 Thesis Structure

On the first pages the motivation and short introduction into the topic of this work is presented. In Chapter 2 related work and the state of the art is presented. First the topic of “human computer interaction” that is relevant for human gesture recognition is discussed. This topic is followed by an overview about the nature of human gestures. For recognition tasks gestures can divided in different categories, mainly concerning their meaning and the way they are performed. This topic is also presented in Chapter 2. An introduction into gesture recognition, an introduction of selected features and a overview on selected classification methods is given in Chapter 2. Chapter 2 is finalized by introducing related works concerning human motion recognition.

In Chapter 3 the methodology for this work is presented. The chapter begins with an introduction of the UCOS sensor used in this thesis. After introducing the sensor a GUI concept to be used with gestures is presented, this section is followed by the introduction of the GUI-related gestures defined for this work. An introduction into the methods for data recording and preprocessing is also part of this section as is the test database and the annotation method used to generate the ground truth. The features calculated from the data generated by the UCOS sensor are presented within this section. Also the machine learning methods used in this work are part of the methodology discussion. Finally the validation method used in this work is presented at the end of Chapter 3.

---

[11] Graphical User Interface

In Chapter 4 the implementation oft the methodologies presented earlier is described. The most important functions used for the training and the evaluation of the classifiers is presented in this chapter.

Chapter 5 begins with a visual evaluation of the calculated features used for gesture recognition. Afther this a preface for classification concerning the test databases is given. This preface includes an overview of the actual contents of the test database generated for this work. Then the results for the classification with Hidden Markov Models are presented, which is followed by the results for the gesture recognition using a Decision Tree. Chapter 5 is finalized with a comparison of the results for both methods.

On the last pages of this thesis, in Chapter 6, a summary of the results from this work is given and also a outlook into the future is presented. Three appendices containing additional information are also added to this work.

CHAPTER 2

Related Work and State of the Art

This chapter gives an overview of related work on gestures, human computer interaction and gesture recognition. Section 2.1 introduces the term “human computer interaction”, which is relevant for human gesture recognition in context with interaction with machines. Section 2.2 gives a brief overview on the nature of human gestures and introduces concepts to categorize gestures. A brief and general summary about the topic of gesture recognition, the typical steps necessary for gesture recognition, an introduction of selected features and a brief overview on classification methods are provided in Section 2.3. In Section 2.4 related work on gesture recognition is presented - two works that are using data acquired with a similar sensor as UCOS and other works that primary intend to recognize dynamic gestures.

2.1 Human Computer Interaction

This section gives a short overview on human computer interaction. It is linked to the same topic discussed in [38]. The main intention of HCI[1] is to analyze interactions between humans and machines, especially computers. Figure 2.1 illustrates that knowledge of different areas of research have to be linked together. Therefore HCI is an interdisciplinary field of science. Figure 2.1 shows that experts from diverse fields of knowledge like human sciences, technology and interface design have to cooperate to make the work with computers (and machines) as comfortable and practical as possible [21]. As the following historic example shows, one may never underestimate the importance of HCI. The primary cause for the accident in the nuclear power plant on Three Mile Island (USA) was poor user interface design: the overload of information displayed on a large number of gauges and displays made it impossible for the operators to find the relevant error messages which could prevent the disaster [89]. This shows that in critical areas, where accidents have a high impact on the environment (e.g. nuclear pollution), HCI becomes important to rule out human error [38] [89]. HCI is not only important in critical applications. It is essential in all areas, where humans and computers have to work together [48] [38].

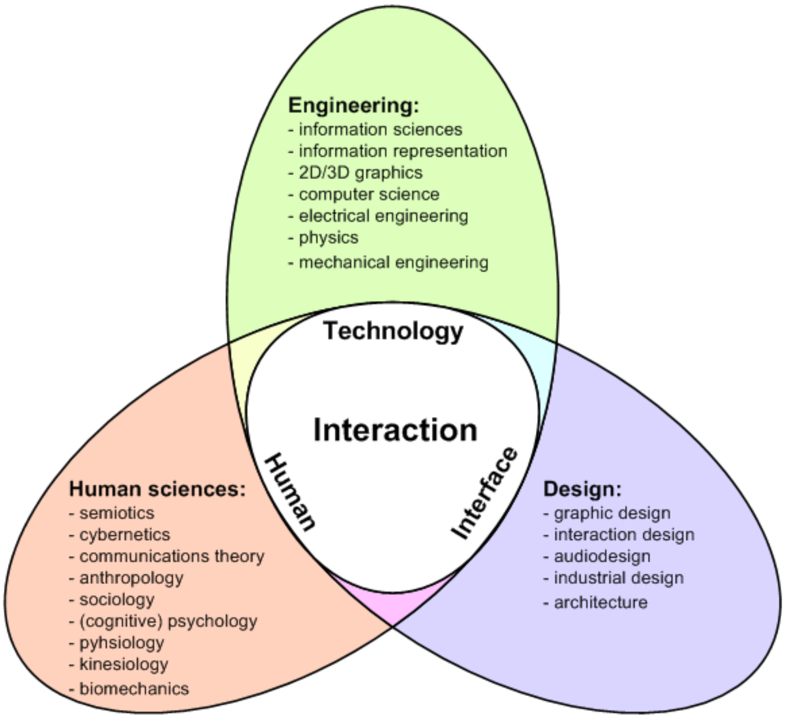


Figure 2.1: The interdisciplinary field of research of HCI according to [21]

---

[1] Human Computer Interaction

Interdisciplinary research combining human sciences, technology and interface design has lead to a better understanding how humans work with computers and machines [38]. According to Norman [34] the interaction between humans and computers can be modeled by the *seven stages of action* illustrated in Figure 2.2. The model visualizes the steps humans take when interacting with computer systems. It describes how users of a computer system form goals and undertake the steps required to achieve the goal of using a computer. The seven stages of action describe the problem experts (writers, designers and engineers) face when they have to meet the goals of the users as two gulfs between the user and the system (developed by the experts). The *gulf of execution* represents the users difficulty to translate a psychological goal into a physical action. The user's difficulty in evaluating whether the response of the computer system meets the desired goal is represented by the *gulf of evaluation*. For a successful interaction between humans and machines it is highly important to build a bridge between the physical and the mental world [34]. [38]

To think about HCI is also useful for gesture recognition intended to control computers. Especially when the gesture patterns are predefined and cannot be changed by the users, intuitive gestures have to be used [79]. In the state of development of a gesture recognition system research (like [20]) has to be conducted, to find out, what gestures are intuitive for the users [79]. This has to be done to reduce the problems appearing in the gulf of execution to a minimum.

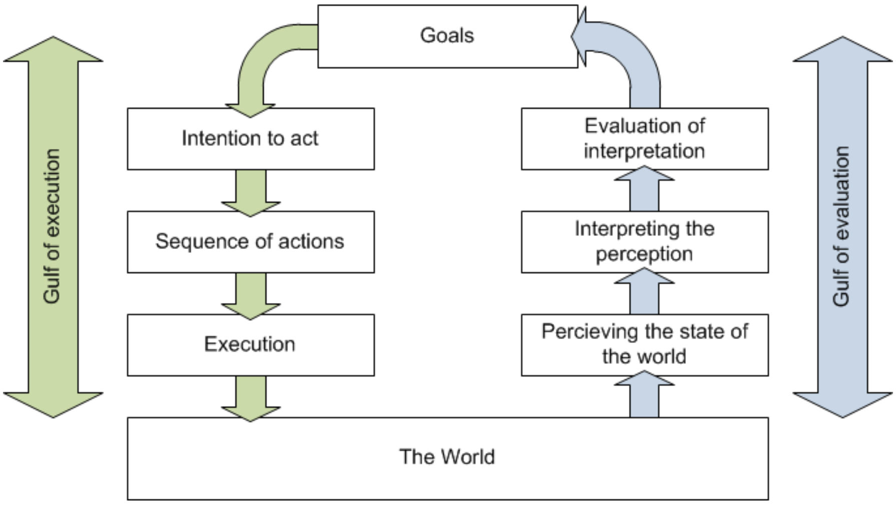


Figure 2.2: Model of interaction according to [21]

2.2 Gestures

For the development of a system for gesture recognition it is necessary to know what is to be recognized. Therefore, in this section an overview on the topic “gestures” is presented. The first part of this section deals with the meaning of gestures for human communication. The second part of this sections introduces how to differentiate and categorize human gestures.

Gestures and Humans

One of the first means of communication used by children are gestures as the usage of verbal means of communication emerges after gesticulation in human development [39]. Gestures and body language play a natural role in the communication of grown ups [39]. For example politicians adjust their body language to to emphasize their message in order to win voters [55]. Gesture forms an integrated system with speech and strengthens the message transported conveyed with words [39]. According to [39] listeners are more likely - how much more is not told by [39] - to grasp what was said in a speech if it is accompanied by a gesture conveying the same message as the words said. Vice versa, a gesture contrary to the words can lead to misunderstandings, making the effect worse than using no gestures at all [39]. There is also considerable evidence, that gestures do not only play an important role for listeners, but they are also important for the speaker [39]. Fewer mistakes and verbal hesitations occur when speakers support their talk by gestures while suppressing gestures leads to a less fluent speech [39]. Gestures are used to support verbal communication even when communication partners do not see each other, for example, when using the telephone or an intercom system [39]. Even speakers who are blind from birth and therefore have never seen somebody moving their hands or amrs. They even do so, using gestures, when they are talking to other blind people. [38] [39] [18] [66]

It is not clearly defined, what gestures have to look like. As mentioned before different cultures are full of varying ways of communication by gestures and body language [62]. If one thinks about dances and rituals of indigenous people one can see that they are full of gestures [30] [27]. This is another prove, of how deep communication through the body and through movement of body parts is embedded within human beings. Gestures can have different meanings, depending on current situations, or on cultural background [38]. For example the well-known hand-gesture divers use to signal “OK” (connecting the thumb and forefinger in to a circle, and holding the other fingers straight or relaxed in the air) has other meanings too. In some countries this gesture symbolizes a number, in other ones it has a insulting meaning [91]. This cultural aspects have to be considered when developing a gesture recognition system, especially when dealing with the development of a gesture set [62].

Types of Gestures

For the development of a system for gesture recognition it is also necessary to take a look at how gestures, especially hand and arm gestures can be differentiated [38]. Generally gestures are understood as expressive, meaningful body motions that involve physical movements of the fingers, hands, arms, head, face, or the whole body [62]. The intention of such a gesture is interaction with the environment or, to convey meaningful information [39]. Gestures are composed of small subspace of possible human motion, mostly specialized to some distinct body parts (hand, face, ... [62]) but for some applications (fitness training [42], gaming [12] ,...) it can be interesting to see the whole body as gesture source. So, a gesture can be seen as a compression technique for the information to be transmitted, that subsequently has to be reconstructed by the receiver [62].

First of all one has to differentiate between static and dynamic gestures, because they are significantly different [38]. Static gestures, also known as poses, have to keep the same shape until they are recognized. For example, they can be differentiated by position, rotation and form of the body part used to gesture. Typical examples for static gestures can be found in sign language, or the way a policemen controls the flow of traffic on the streets. It is not the way a posture is gained that determines the meaning of a static gesture, but the information is encoded in the posture itself. In comparison to that a single movement contains the major information for dynamic gestures. Gestures that encode their information in both movement and shape are called dynamic gestures too. Examples for dynamic gestures include pointing gestures and other gestures that encode information in movement (e.g. German sign language [56]) [38]. Due to the technological aspects of the UCOS sensor, which are explained in Section 3.1, only dynamic gestures can be recognized by the sensor and are therefore used in this work [13].

As gestures cannot only be categorized by the amplitude (or lack) of the movement necessary to shape them, two different taxonomies concerning gestures are reviewed on the next pages: a general taxonomy for HCI according to [47] and a taxonomy specialized for hand/arm gestures according to [70]. A major problem within gesture research is the lack of commonly used terms describing the interactions [47]. A taxonomy which sorts gestures into five categories is introduced in [47]: deictic gestures, manipulative gestures, semaphoric gestures, gesticulation and language gestures. This taxonomy is illustrated in Figure 2.3. The main idea behind this categorization is to define the optimal gesture class for each interaction scenario possible in HCI.

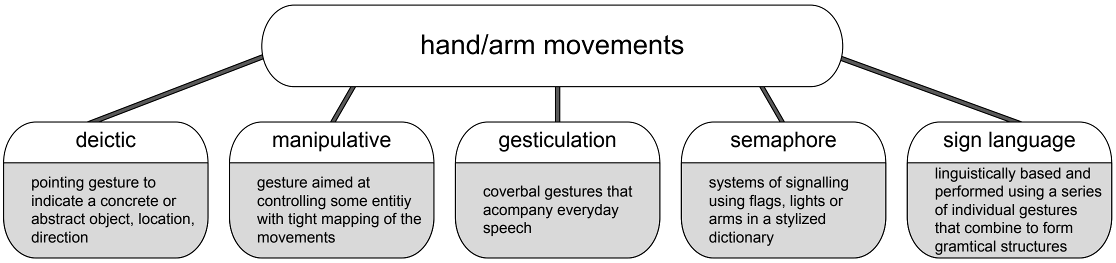

hand/arm movements
deictic
pointing gesture to indicate a concrete or abstract object, location, direction
manipulative
gesture aimed at controlling some entitiy with tight mapping of the movements
gesticulation
coverbal gestures that acompany everyday speech
semaphore
systems of signalling using flags, lights or arms in a stylized dictionary
sign language
linguistically based and performed using a series of individual gestures that combine to form gramtical structures

Figure 2.3: Taxonomy of gestures in HCI according to [47]

- **Deictic gestures** primary consist of pointing actions that enable the selection or identification of an object. [40] describes deictic gestures as pointing gestures that refer to people, objects, or events in space or time. The content of a speech is modified when accompanied by this type of gestures as they help with disambiguation. Deictic gestures can be compared with the role of spoken spoken intonation. These gestures represent a very basic form of communication and are often intuitively used.
- **Manipulative gestures** are defined by the direct relation between hand/arm movement and the movement of a controlled entity or virtual object. In [72] manipulative gestures are defined as: “those whose intended purpose is to control some entity by applying a tight relationship between the actual movements of the gesturing hand/arm with the entity being manipulated”.
- **Semaphoric gestures** are defined as systems of signaling using flags, lights or arms [72]. They can consist of dynamic and static gestures, wheras static gestures are represented by a defined body shape or posture. The main task of semaphoric gestures is to communicate symbols to a machine. Each of the semaphoric gestures resembles a symbol. An example for a static semaphoric gesture (or symbol) is the “OK” gesture mentioned earlier, an example for a dynamic semaphoric gesture is implying a “Hello!” by waving the hand. The gestures defined for this work can be sorted into this category.
- **Gesticulation** is one of the most natural forms of gesture. It is an essential part in dialogues between humans that normally consist of speech and gestures [47]. Gesticulation is a specific and spontaneous hand or arm movement when speaking. It does not require any training or learning. The poise of the talking is not relevant for this type of gesture.
- **Language gestures** have to be discussed independently from other gesture styles. They are linguistic-based and they require the continuous interpretation of multiple and individual hand signs that have to be combined to form grammatical structures [47]. An additional level of processing is required to understand language gestures, because a whole collection of gestures has to be interpreted to understand the information encoded within them. In sign language gestures are not only semaphoric, but also deictic. Even copying of gestures (mimetic gestures) can be found in sign language [38]. In HCI sign languages are important for assistant and teaching systems, e.g. when teaching sign language to children.

The taxonomy by [47], that is introduced above, does not only concern hand/arm movements, but it also refers to the meaning of gestures where the whole body is involved. A taxonomy that is more specialized on hand/arm gestures concerning HCI introduced in [70]. The tree in Figure 2.4 illustrates the division of hand/arm gestures into different classes. The classification begins with splitting hand/arm movements into two categories: Unintentional movements, which have no meaning for HCI and gestures that fulfill a specific function. These gestures can be split up into two subgroups: manipulative gestures and communicative gestures. As discussed earlier, manipulative gestures refer directly to an object and are not divided into further subclasses. Communicative gestures can be divided into acts and symbols. Symbols either can be referential like moving a finger according to the movement of a wheel, or the modalize to strengthen the meaning of spoken instructions. Acts can be divided into deictic gestures, usually represented by pointing actions and mimetic gestures that imitate known actions. The gestures that are defined for this work in Section 2.2 are designed to control the multimedia interface introduced in Section 3.2. Therefore deictic and symbols are used in this thesis.

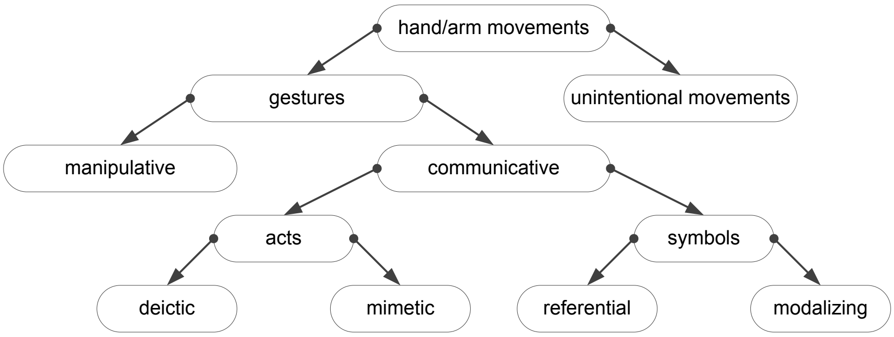


Figure 2.4: Taxonomy of hand/arm gestures in HCI according to [70]

2.3 Gesture Recognition

In this section information about gesture recognition are presented. The first part of this section introduces gesture recognition in general and gives an overview about the topic. In the second part of this section the typical processing steps used in gesture recognition are described. This description can be seen as a summary of all works that have been investigated for this study. In the last part of this section a selection of classification methods used for gesture recognition is introduced.

Overview

According to [79], [24] and [62] gesture recognition is a topic of computer science. Its goal is to interpret human gestures via mathematical algorithms. As described earlier gestures can have different meanings. So a gesture recognition system is dependent on the source of gesture and the application it should be used for [62].

According to [79] and [62] gesture recognition has a wide-range of applications. Thus, it does not only act as an input source for computer applications. Examples for applications for gesture recognition are: developing aids for the hearing impaired, recognizing sing language, monitoring the emotional state of people, lie detection, automated fitness evaluation for (elderly) persons, fall detection, medically monitoring of stress and/or emotion, lie detection, robotics, applications in the automotive sector (e.g. to detect if the driver is tired) and gaming [66] [62] [79] [49] [35]. Gesture recognition, especially understanding gestures, plays a role in robotics, where the recognition of the meaning of movements of human beings turns out to be critically important in order to improve HRI[2] [66]. To distinguish the different meanings of human movements is challenging, as equal actions and movements can have different meanings depending on the current situation and on social and cultural context [66] [62]. This makes it challenging to develop independent systems that are able to understand gestures.

Current research on hand gesture recognition for HCI relies on mathematical models like HMM or NN[3] [62]. In addition particle filtering in combination with the condensation algorithm [44], finite state machines and soft computing approaches are used for gesture recognition [62]. It is also possible to use 3D[4] body pose estimation and skeletal models for gesture recognition [87]. Different devices are used to gather data on human motion (or the motion of limbs) and/or static positions, too. Therefore the numerous approaches to gesture recognition do not only differ in the way how the movement or position of humans is statistically evaluated, but also in the way the movement or position of humans is acquired. Some works use a wearable device, such as data gloves [69] or suits equipped with sensors [46]. Other devices used for gesture recognition are for example mice and special remotes like the WiiMote [38]. Microsoft research introduced a solution which enables the recognition of acoustic signals produced by a mobile phone, which was moved around in a room equipped with microphones, for gesture recognition [84]. Studies on gesture recognition rely on different computer vision technologies [62]. For example stereo camera systems generating 3D data are used for gesture recognition as are standard consumer WebCams [62] [87] [46] [28]. Active sensors (like Microsoft Kinect) that project a pattern of light and recognize the deformation of the pattern, are used as are time-of-flight cameras [26] [41] [62] [28] [51]. Even combinations of different vision sensors are used for gesture recognition - in [28] the Kinect sensor is used in combination with a web cam and in [33] time-of-flight and color cameras are used together. The visual approaches for gesture recognition can also be divided into approaches using markers on persons and such that do not use markers [38]. Since the introduction of Microsofts Kinect sensor in 2010, the field of vision based gesture recognition has been noticed by moving into consumer market [12] [10] [79]. The sensor was intended primary as a game controller for Microsofts XBox 360 [12], but shortly after the release the technology inspired people all over the world to develop other applications for the sensor [5], even including the control of PC[5]s [14]. Finally, an official software development kit for using the Kinect sensor with the PC was released by Microsoft [6] making further developments possible [79]. The idea to use the visual recognition of human motion as input for HCI, especially as game controller, was also used in 2003, when Sony introduced the EyeToy as controller for the PlayStation [79]. Nowadays, in 2012, even TV-sets with gesture recognition are available on the markets as Samsung has introduced the Smart-TV that can be controlled by hand gestures [10]. On the left side of Figure 2.5 Samsung's Smart-TV is shown. The right side of Figure 2.5 shows the usage of the Kinect sensor for controlling games.

---

[2] Human Robot Interaction

[3] Neuronal Networks

[4] Three-Dimensional space

[5] Personal Computer

Figure 2.5: Commercial applications for gesture recognition: Samsung Smart-TV and Microsoft XBox 360 + Kinect (images from [10] and [12])

Typical Processing Steps in Gesture Recognition

The typical processing steps for gesture recognition are: data acquisition, segmentation and/or preprocessing, feature extraction and classification [38]. This chain of steps is illustrated in Figure 2.6. As stated earlier, works concerning gesture recognition use many different types of sensors to acquire the position or motion of humans [62]. Depending on the method the motion data is gained, different preprocessing methods have to be used: In works using Kinect [77] [16] or time-of-flight cameras [25] [26] the gathered depth information is used to segment the body or body parts like hand, arm, leg and head out of the recorded data. If a mono camera is used, algorithms like the Condensation algorithm introduced in [44] are used to recognize body parts of interest [76]. The segmentation is performed to keep only objects of interest [90]. The objects of interest are used to calculate features for each gesture [79].

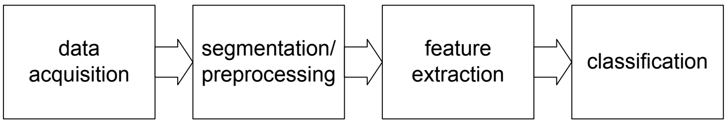


Figure 2.6: Data processing steps used in gesture recognition (image adapted from [38])

Features used for gesture recognition vary from relatively simple features like “velocity of the tracked hand” [56], “distance between hands” [56], “absolute position of the hand” [56], “highest point of motion” [67], “median of the height of a moving person” [67], “orientation of the main axis of a moving person” [67] and “vertical velocity of the highest point of a moving person” [67] to more complex features like “relative pixelcount“ [93], “shape-context of a human” [42], “gradient histogram” [29] and “the overall depth information” [90]. As one can see, many different observations and characteristics can be used as features - the selection of features depends on the data acquired and the intended application of the gesture recognition. A selection of features from related work that were introduced above is presented in the next section. 2.3. These features are used for the training of a classifier with ground-truth data and for the recognition of untrained samples. The section after the introduction of the features gives an overview on classification methods that were used in related works.

Features

A fundamental task in the means of gesture recognition is the selection of the best features for the classification [42]. Redundant features cause unnecessary computation load and the selection of poor features likely leads to false classifications [42]. Features that describe the content at the best possible rate have to be selected [42]. In this section, selected features (relative pixel count, depth features and 2D features) that were used in related work concerning human motion recognition, are described. Some of the presented features make use of 3D information and others are using image-processing techniques. Most of the features described here have already been used as input for Hidden Markov Models as for other classification methods.

Relative Pixel Count

In [42] and [93] a so-called “relative pixel count” is used as input for a HMM classification. In [93] binary images are generated from conventional video information. In this binary images a region of interest around the human body was selected. This region is divided into meshes and for each of the meshes the relative count of black pixels is calculated [42]. The feature vector is built with Equation 2.1 [93]. In the equation the ratio of black pixels in each mesh is calculated. M_m, N_m are the counts and i, j are the indexes along the image dimension [93]. In Figure 2.7 the complete mesh feature generation is illustrated.

$$f(i,j) = \frac{\text{number of black pixels}(i,j)}{M_m \cdot N_m} \tag{2.1}$$

In [93] reconnection rates of 96% for the detection of 5 different tennis strokes were achieved by using only this feature. In [42] the activated pixels delivered from the sensor were used to calculate the relative pixel count feature on on the fly. In combination with another feature recognition rates of 97.33% were achieved in [42].

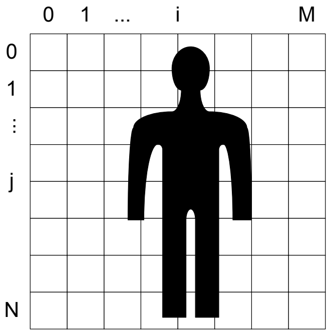


Figure 2.7: Mesh feature (graphic adapted from [93])

Features based on Depth Information

A special advantage available with 3D systems is the option to use the depth information to calculate features [42]. The depth information can be used for calculating the relative distance of certain body parts, or the whole body, to the sensor. In [90] preprocessed depth images were used to calculate these three depth based features:

- Body centroid (2D)

$$x_s^b = \frac{\sum\limits_{n=1}^{N} x_n^s 1 \left\{x_d^n \in \text{Depth level of body}\right\}}{\sum\limits_{n=1}^{N} 1 \left\{x_d^n \in \text{Depth level of body}\right\}}$$

- Hand displacement (2D)

$$x_s^h = x_s^b - \frac{\sum\limits_{n=1}^{N} x_n^s 1 \left\{x_d^n \in \text{Depth level of hand}\right\}}{\sum\limits_{n=1}^{N} 1 \left\{x_d^n \in \text{Depth level of hand}\right\}}$$

- Relative depth level of the hand (1D)

$$d_r^h = x_d^b - x_d^h$$

  x_d^b and x_d^h are the depth levels (grayscale values in the depth map images) of the body and the hand [90].

In [90] recognition rates around 92% were achieved when these features and HMMs were used for the detection of nine different gestures for gaming applications.
In [67] a 3D sensor is used, which only recognizes motions is used. In Figure 2.8 two frames of data from such a sensor can be seen [67]. Information on the basic features computed from that depth is visible in this figure, too. The so gathered basic features a reused as feature themselves or used to calculate further combinations of features. Thus, a set of time dependent features is collected:

- Angle of the main axis
- Vertical volume distribution ratio
- Vertical velocity of the highest point
- Orientation of the main axis
- Ratio of the occupied ground area to height
- Highest point

In [67] this features lead to promising results when they were used on HMMs and on a MLP[6] to recognize falls of humans.

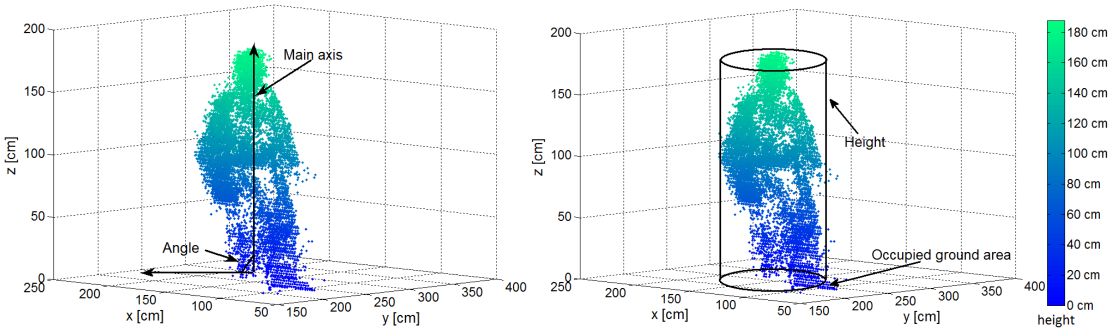


Figure 2.8: Features used in combination with a motion sensor (figures from [67])

Geometric Information 2D

In [80] trajectories generated with a computer mouse are used as gesture source. In Figure 2.9 a gesture from [80] is shown with its relevant lengths and angles labeled with the intermediate variables. By using this basic geometric informations, a total of thirteen geometric features are computed [80]. Those features are interesting for this work and are therefore introduced here. The features introduced in [80] are:

- Cosine and sine of initial angle with respect to the X axis:

$$d = \sqrt{(x_2 - x_0)^2 + (y_2 - y_0)^2} \qquad f_1 = cos\alpha = \frac{(x_2 - y_0)}{d} \qquad f_2 = sin\alpha = \frac{(y_2 - y_0)}{d}$$

- Length of the bounding box diagonal:

$$f_3 = \sqrt{(x_{max} - x_{min})^2 + (y_{max} - y_{min})^2}$$

- Angle of the bounding box:

$$f_4 = \arctan \frac{x_{max} - x_{min}}{y_{max} - y_{min}}$$

- Distance between the first and the last point:

$$f_5 = \sqrt{(x_{p-1} - x_0)^2 + (y_{p-1} - y_0)^2}$$

- Cosine and sine of the angle between the first and the last point:

$$f_6 = cos\beta = \frac{(x_{p-1} - y_0)}{f_5} \qquad f_7 = sin\beta = \frac{(y_{p-1} - y_0)}{f_5}$$

---

[6] MuliLayer Perceptron, an artificial neural network

- Total gesture length:

$$\Delta x_p = x_{p+1} - x_p \qquad \Delta y_p = y_{p+1} - y_p \qquad f_8 = \sum_{p=0}^{p-2} \sqrt{\Delta x_p^2 + \Delta y_p^2}$$

- Total angle traversed:

$$\Theta_p = \arctan \frac{\Delta x_p \Delta y_{p-1} - \Delta x_{p-1} \Delta y_p}{\Delta x_p \Delta x_{p-1} - \Delta y_p \Delta y_{p-1}}$$

$$f_9 = \sum_{p=1}^{p-2} \Theta_p \qquad f_{10} = \sum_{p=1}^{p-2} |\Theta_p| \qquad f_{11} = \sum_{p=1}^{p-2} \Theta_p^2$$

- Maximum speed (squared):

$$\Delta t_p = t_{p+1} - t_p \qquad f_{12} = \max_{p=0}^{p-2} \frac{\Delta x_p^2 + \Delta y_p^2}{\Delta t_p^2}$$

- Path duration:

$$f_{13} = t_{p-1} - t_0$$

According to [80] some of the features (f_1, f_2, f_6, f_7) are sines or cosines of angles. Others $(f_5, f_{10}, f_{11}, f_{12})$ depend on angles directly and require inverse trigonometric functions to compute. A four quadrant arctangent is needed to compute Θ_p that returns an angle between $-\pi$ and π. These features are used on elementary statistical pattern recognition techniques [80] where recognition accuracies around 97.00% were reached.

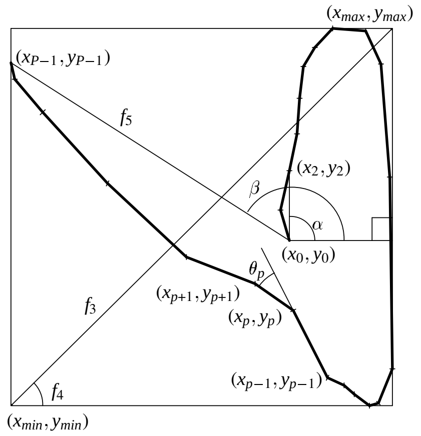


Figure 2.9: Geometrical features calculated from trajectories (from [80])

Classification Methods

This section gives a short introduction into classification methods used for gesture recognition. In some of the related works about gesture recognition and human motion recognition machine-learning methods like NN, HMM and SVM are used. Other works use classical pattern recognition methods like pattern matching. On the following pages selected recognition methods - like NNs, SVMs and HMMs - are briefly introduced. Information about useful literature containing further details on the methods is presented here, too.

Neuronal Networks

In [42] Neuronal Networks (NN) are introduced as method for gesture recognition. [42] states, that NN were first used for gesture recognition in the early 90's by [63]. Subsequently they were adapted for further applications [42]. According to [62], NNs are becoming more important in the area of gesture recognition, especially, in static gesture recognition tasks. According to [42], the main application of NN for classifying human motion can be seen in the recognition of sign language such as in [63]. In [67] the MLP, which is a form of NN, as used to detect human falls. In this work the also evaluated HMM led to slightly better detection results than the MLP. According to [88], examples for other applications for NNs are: recognition of speakers in communications, diagnosis of hepatitis, recovery of telecommunications from faulty software, interpretation of multimeaning Chinese words, undersea mine detection, texture analysis, three-dimensional object recognition, hand-written word recognition and facial recognition.

Generally, the information processing paradigm known as NN is inspired by the way how biological nervous systems, such as the brain, process information [88]. In NN neurons are processing information to solve certain problems [42]. Like the biological role model, NNs are learning by example [88]. Thus, NNs are trained for specific applications such as texture analysis [88]. In Figure 2.10 a simple neuron, with multiple inputs and just one output, is illustrated.

According to [42] NNs are not the first choice for the recognition of dynamic gestures. Therfore, NNs are not explained in detail here. However, a detailed description of NN can be found in [88].

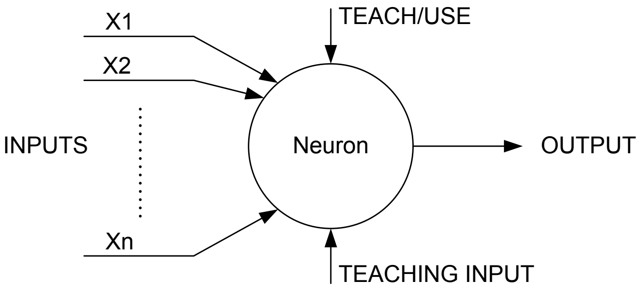


Figure 2.10: A simple neuron (image adapted from [88])

Support Vector Machines

According to [42], Support Vector Machines (SVM) are also used for gesture recognition. For example in [81], SVMs are used to recognize human actions and in [32] SVMs are used to recognize pointing gestures. SVMs offer outstanding generalization capability and have a reputation of being a highly accurate paradigm [81]. According to [42], SVMs were first introduced by [31] for pattern recognition. Originally, SVM models were defined for the classification of linearly separable classes of objects [45]. Basically, a SVM finds the unique hyperplane having the maximum margin for any particular set of two-class objects [45].The optimal hyperplane and the optimal margin for such a linear separable case are shown in Figure 2.11. In this figure the margin of the largest separation between the two classes is represented by the gray marked boxes [31]. According to [45], classes that cannot be separated with a linear classifier can be classified with SVM. If non-linear classes are to be classified, the coordinates of the objects are mapped into a high-dimensional features space in which the two classes can be separated with a linear classifier [45]. More details on SVM in general, and on their applications can be found in [45], [31] and in [11].

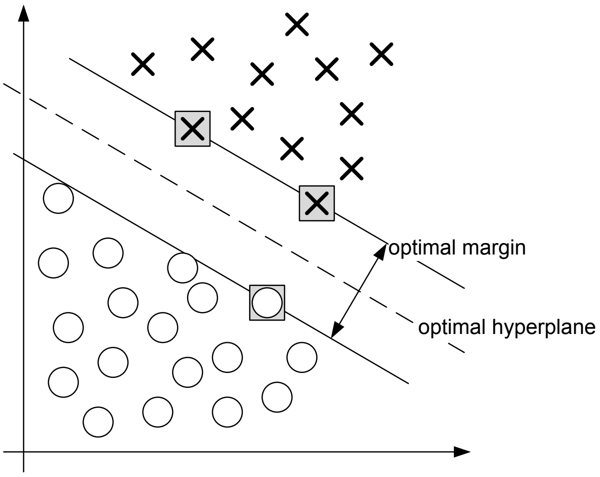


Figure 2.11: SVM: A linear separable example (figure adapted from [31])

Hidden Markov Models

The Hidden Markov Model (HMM) is a stochastic model [75] that has a lot of applications - for example: automatic speech recognition [75], hand-writing recognition [19], spam filters [58]. In [90], [93], [37], [29], [35], [67] and [42] HMMs are used for the classification of human motion, respectively gestures.

According to [62], the HMM is a double stochastic process governed by an underlying Markov chain that has a finite number of states and a set of random functions, where each of the functions is associated with one state [75] [36]. This stochastic process is started in one of the states. In discrete time instants an observation symbol according to the random function corresponding to the current state is generated [62]. For each transition between the states a pair of probabilities does exist [62]. These probabilities are defined as follows:

1. “transition probability, which provides the probability for undergoing the transition;” [62]
2. “output probability, which defines the conditional probability of emitting an output symbol from a finite alphabet when given a state.” [62]

According to [62], the HMM is rich in mathematical structures. Spatio–temporal information is efficiently modeled by the HMM [62]. As the only thing visible in this process is a sequence of observations the model is termed “hidden” [62]. In all recognition processes which use the HMM, a HMM model has to be constructed for each event that has to be classified [62]. HMM based gesture recognition shows promising results when applied on AE data [67] [42]. More information about HMMs can be found in [75], [36] and on [85]. Additionally, more detailed information on HMMs is presented in Section 3.7 of this work.

2.4 Works on Gesture and Motion Recognition

In this section selected works concerning visual gesture recognition and visual motion recognition are presented. In the first part of this section works using a sensor closely related to the sensor used in this work are presented. Works using other sensors and methods for visual gesture recognition and visual motion recognition are presented in the second part of this section. The last part of this section contains a summary of the introduced works and compares the results of these works with other works that are not introduced in detail in this study.

Address Event Data used for Motion Recognition

Two works that use AER data generated by sensors of the UCOS family for fall detection [67] and for analyzing dance movements for fitness training of elderly people [42] are introduced here. Both, [67] and [42] use a further evolution of AIT’s dynamic stereo vision sensor UCOS - the so called ATIS[7] sensor. The main difference between the ATIS and the UCOS sensor is, that the first one has a higher resolution (304 x 240 pixels) than the latter one (128 x 128 pixels), but the basic principle of the devices are the same. The UCOS sensor is described in detail in Section 3.1. The AE analyzed in this two works are not directly gestures but they can be seen as such. The event of “falling down” can be understood as a body gesture while the dance movements resemble gestures that involve the whole body or at least parts of it. Both of works are part of projects belonging the AAL joint program.

Fall Detection

According to the research conducted for [67] the number of elderly people in the population is increasing. Therefore further technical development of minimal intrusive assistance systems is

---

[7] Asynchronous Time based Image Sensor

necessary to enable elderly people to live as long as possible on their own without considerable help from other people. Thus a major issue for unsupervised living of elderly people is the recognition of critical situations (especially falls), which should trigger an alarm automatically for getting help, if the person cannot solve the incident alone. The improvements in this field will not only to reduce costs for care facilities, but it will also improve the quality of live for elderly people [3]. Therefore the development of such systems is a major goal of the project CARE (see [3]). To evaluate the abilities of the ATIS sensor when used for of fall detection a database containing 7568 samples without falls and 113 sample with falls was generated. The falls were recorded in a laboratory, which was equipped to emulate a typical home environment as displayed in Figure 2.12. A total of ten persons, seven men and three women, acted on performed activities like “sitting on a chair”, “standing up or sitting down to a chair”, “picking up objects from the floor”, “walking around slowly”, “lying down to a couch” and other non-critical actions. The test persons also performed critical situations which all included different fall scenarios like “falling backwards because of loosing balance” or “falling because of missing the chair to sit on”. A sequence of such a fall scenario is illustrated in Figure 2.13. In the beginning, the person is walking and at the end the person lies on the floor with little movement.

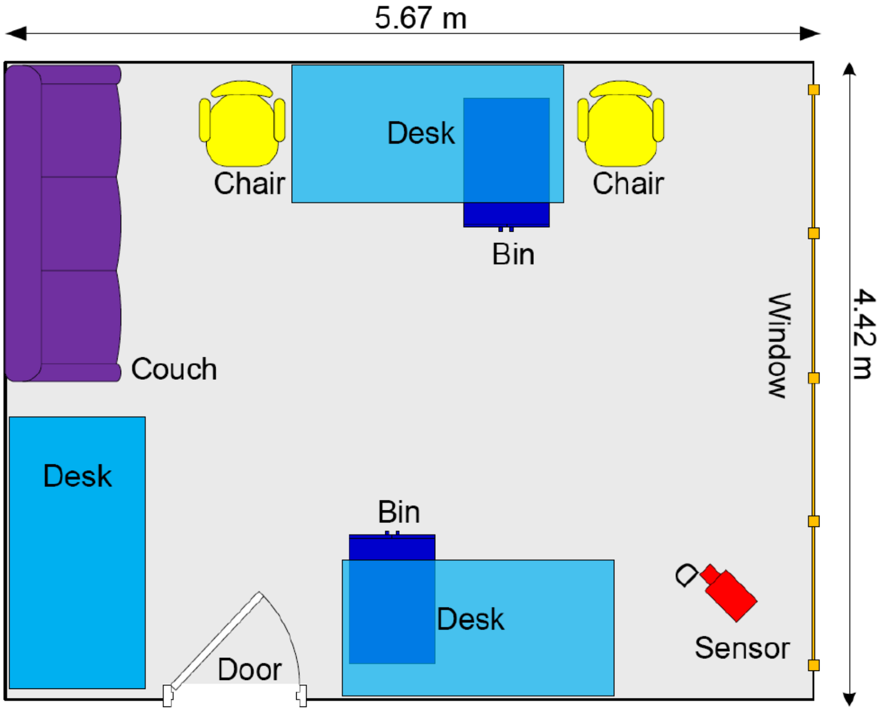


Figure 2.12: Laboratory environment for the recording of the acted scenarios (from [67])

The previously recorded data was used to extract different features like “orientation of the main axis” and “vertical volume distribution ratio” (see Section 2.3 and [67] for more details). The features were then used to train and evaluate two different machine learning approaches, HMM and MLP. Using a cross validation on the HMM, 109 out of 113 fall samples were classified correctly but there were also 100 out of 7568 non fall samples, which were classified as falls. The classification using the MLP led to a similar result. Out of 113 fall samples 110 were classified correctly, but 103 out of 7568 non fall samples were also recognized as falls. Basically the results from [67] show, that the ATIS sensor in combination with HMM or MLP could be used for fall detection. [67] states, that the rate of misclassified non fall events is a major issue for further development, since the false classifications would lead to a false alarms keeping emergency responders busy.

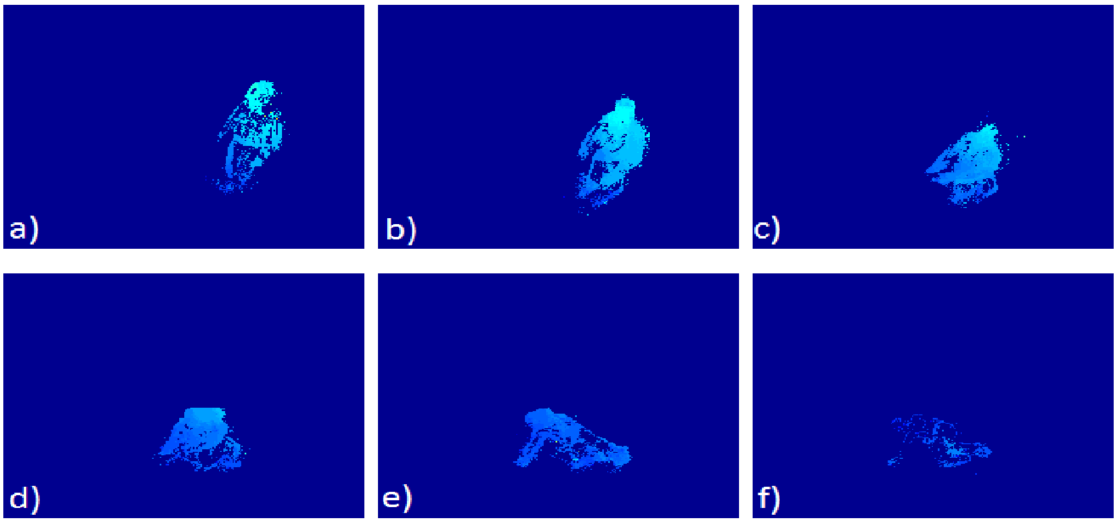


Figure 2.13: Snap-shots of the depth map during the fall. The depth is color coded in a way that brighter green colors stand for nearer objects. (from [67])

Dance Fitness Training

The work presented here is part of the Silvergame project (see [86]). The motivation for [42] and the Silvergame project is also related to the growing number of elderly people in the population. Research conducted for [86] and [42] showed that it is important (not only, but especially) for elderly people to make regular exercises to stay healthy and vital. Therefore a system that supports the fitness training of elderly people has been developed, which can provide this target group with facilities so they can stay healthy. Some elderly people do not have the opportunity to go to private trainers, a system that recognizes human motion in the area of dance and fitness training for elderly people is introduced in [42]. This system is meant to encourage elderly people to do their regular dance exercises and to give feedback on the quality of the training. A dance exercise consists of a number of different figures or activities depending on the choreography and the piece of music. For the work introduced here eight different activities like “arms pointing with 360 degree axis left rotation” and “arms pointing with 360 degree axis right rotation” were used (see [42] for more details). One of the activities is displayed in Figure 2.14.

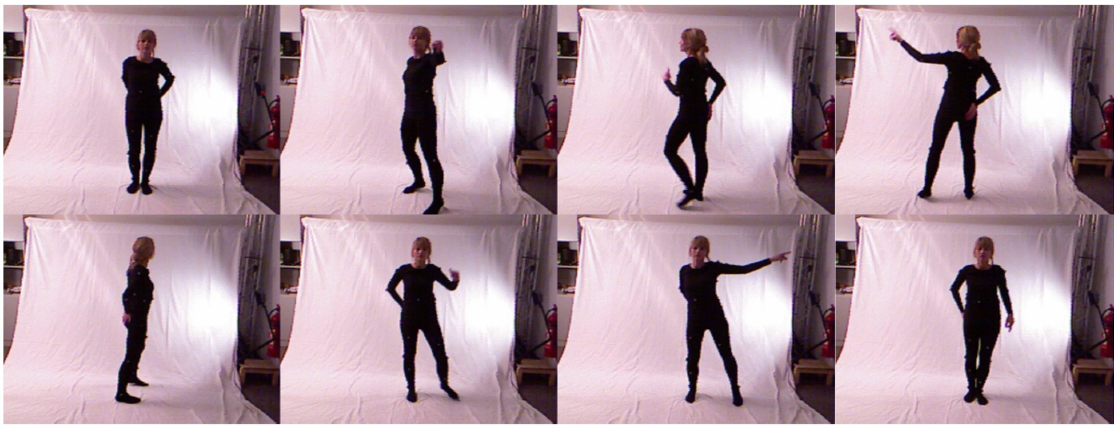


Figure 2.14: Dance activity, a person performing a rotation to the right and pointing with the hand. (from [42])

Activities	**A1**	**A2**	**A3**	**A4**	**A5**	**A6**	**A7**	**A8**	**Recog. Rate[%]**
A1	77	0	0	0	0	0	0	0	100.00
A2	0	80	0	0	0	0	0	1	98.77
A3	0	2	66	0	0	0	0	4	91.67
A4	0	0	0	78	0	0	0	2	97.50
A5	0	0	0	0	29	0	0	0	100.00
A6	0	1	1	0	0	86	0	0	95.56
A7	0	0	0	0	1	0	79	3	95.18
A8	0	0	0	0	0	0	0	68	100.00

Table 2.1: Full evaluation matrix for best result from [42] (A1-A8 represent the activities)

For evaluation and training fifteen individuals carried out eight different activities which were recorded using the ATIS sensor. From this actions not only the *stereo data* containing depth information, but also the *mono data* from one sensor and the *left-right overlay mono data* without depth information were used. These three different types of data were used separately to calculate features like "relative pixel count" and "relative disparity/distance". These features were then used to train a HMM. The cross validation revealed that the usage of *left-right overlay mono data* leads to a slightly better average recognition rate of 97.33% compared to the 95.65% achieved when *stereo data* was used. The full evaluation matrix for the best recognition result is shown in Table 2.1.

Selected Work on Vision-Based Gesture Recognition

In this section three selected works concerning vision-based gesture recognition are introduced. Two of the works use the Microsoft Kinect sensor and one uses a stereo camera. These works are presented here, because their topic is the recognition of dynamic gestures as in this study.

Sign Language Recognition

In [56] the implementation of a framework for gesture recognition using Microsofts Kinect and HMMs is presented. This framework is intended to recognize the signals of the German sign language. Information about the German sign language can be found in [23] and [2]. In [56] the functionality of the Open NI driver [8] was used with the Kinect. This driver is necessary to use the middleware NITE that features a bult-in body segmentation function for the Kinect [7]. This body segmentation supplies the gesture recognition framework with information about the position of the users' joint positions. These join positions are transmitted as three-dimensional vectors containing the coordinates of each body part (e.g. head, left hand, right hand,...). The position informations are used to calculate the features such as "velocity", "distance between hands" and "absolute position of the hand" [56]. The features are used to train a HMM for each gesture to be recognized [56]. Figure 2.15 shows a sequence of recordings that is used to calculate features which are then used to train a HMM.

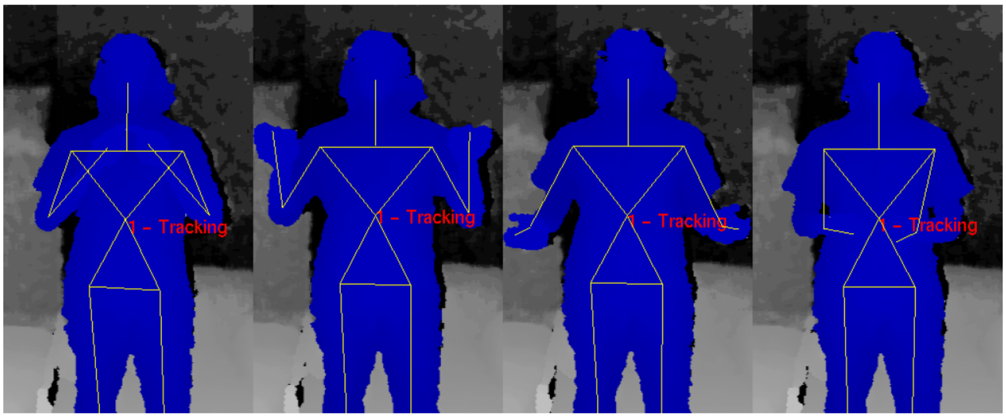


Figure 2.15: Example of a training sequence (from [56])

In Figure 2.16 the dynamic aspects of the German sign language are highlighted with green arrows. In this figure the signs "Giraffe", "Wasser kochen", "Berg" and "Mitte" can be seen. Six additional signs - "Berlin", "Danke", "Kreuzberg", "Neukölln", "Paket" and "Verstehen" - were used for the training and the evaluation of the framework. Thus, a gesture set of ten gestures was used. Informations on how this gestures are performed can be found in [2]. For evaluation a cross validation method was used. Recorded data from one person fluent in sign language and one peson not fluent in sign language were used. One third of the recorded samples was kept and the rest was used to train the HMMs [56]. The data not used for training was then used to evaluate gesture recognition framework. The best results in [56] lead to recognition results of 100% for all signs with exception of the sign "Verstehen", where a accuracy of 97.7% was reached. [56] concludes, that the approach of using depth cameras for sign language recognition is worth further consideration and leaves room for further improvements.

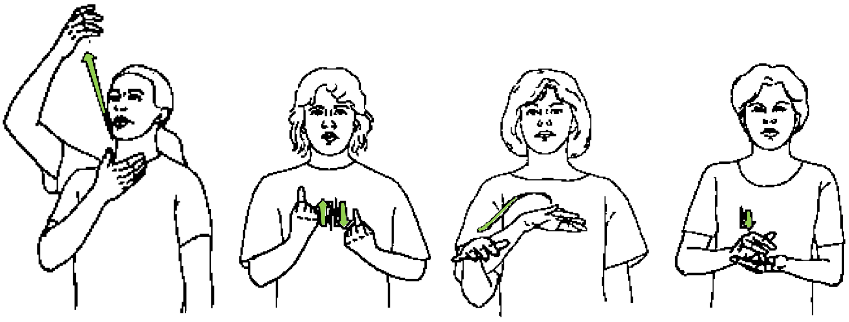


Figure 2.16: The signs “Giraffe”, “Wasser kochen”, “Berg” and “Mitte” in German sign language, motion is marked green (adapted from [2])

Recognition of the Graffiti Alphabet

In [53] the methods developed in a preceding work on recognizing gestures on a tablet device [54] are applied to the hand-tracking results of the Kinect sensor. In the preceding work a “...template-based incremental recognition algorithm for pen strokes and touch-screen gestures” was introduced [54]. As the tracking information gathered by using the Kinect sensor does not differ much from the strokes of a pen on a tablet, the same algorithm can be applied on both with minor changes for the Kinect [53]. Basically, the algorithm estimates the posterior probabilities of the user’s currently incomplete stroke within a set of template classes [54]. Using this algorithm it is possible to predict the user’s intended template gesture based on a partial stroke [54]. An advantage of this method is that continuous feedback from the recognizer can be provided to the user while producing the stroke [54]. For this continuous recognition the predefined templates are divided into segments. Such a segmentation is illustrated on the left side in Figure 2.17. For each of the user’s partial strokes the likelihood of being the best match for one of the templates is calculated [54]. It is also checked whether the user’s stroke matches not only the segments but a complete template. If a stroke describes a complete template, the corresponding template class has to be prioritized higher than those template classes whose best matches are segments representing complete templates [54]. On the right side of Figure 2.17, a partial stroke that matches either one of two templates’ prefix segments is illustrated. Without knowing the end-point of the gesture, the predictions made by the recognition algorithm might fluctuate between both templates [54]. Therefore, using an end-point bias is used to match a complete template when the user finishes the gesture [54]. As similarity measures Euclidean distances and turning angles are used to calculate the similarity between point sequences [54]. More details on the algorithm can be found in [54].

As mentioned earlier, the algorithms developed in [54] are applied to the hand tracking results of the Kinect sensor. To determine the beginning and the end of a gesture, an *input zone* was defined in a certain distance between the user’s hand and the Kinect sensor. If the user is moving the hand in this area, the motion is interpreted as gesture-input [53]. The setup used in [53] is illustrated on the left side of Figure 2.18. To evaluate the gesture recognition accuracy, a set

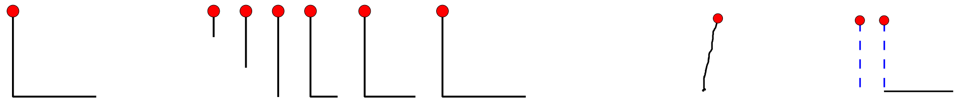


Figure 2.17: On the left: A complete template and the segments generated from it. On the right: an example of two segments matching a user's stroke. (graphics from [54])

of one-handed and two-handed gestures was used. The one-handed gestures used for [53] are illustrated on the right side in Figure 2.18. In total 1683, gestures that were performed by 18 persons have been collected. These data were then used to train and to evaluate the recognition algorithm [53]. The recordings from eight persons were used for the training of the algorithm, the rest was kept for the evaluation. When using the test data on the trained recognition algorithm a maximum accuracy of 92.7% was achieved for one-handed gestures and a maximum accuracy of 96.2% was achieved for two-handed gestures.

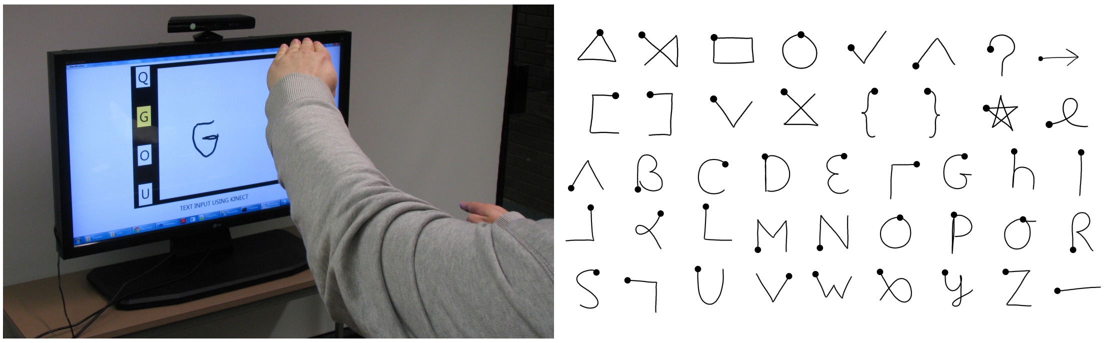


Figure 2.18: On the left: the setup used to recognize the stroke gestures. On the right: set of one handed gesture from the $1 gesture set [92] and the Graffiti alphabet. (images from [53])

Human Body Gestures as Inputs for Gaming

In [90] human body gestures that are recorded with a calibrated stereo camera are intended to be used as input for gaming. The captured stereo pairs are initially corrected for geometric and photometric distortion. Then the correspondence problem between the two images is solved by using the Markov Random Field framework [90]. The stereo images is optimized by using belief propagation [90]. The outcome of this process is a depth (or disparity) map: The depth map measures how far away each pixel is from the camera [90]. This depth map is used to calculate features for gesture recognition. Therefore, some preprocessing is necessary: First the depth map is down-sampled to 32 x 24 pixels. After the down sampling discontinuity-preserving smoothing is allied to the data [90]. In the last preprocessing step the human body is segmented from the disparity data. This preprocessing steps are illustrated on the left side in Figure 2.19. The shape of the human performing the gesture is used to calculate the features. The features used in [90] are: "body centroid (2D)", "hand displacement (2D)" and "relative depth level of the hand (1D)". This features are used to train a HMM for each gesture to be recognized. More details on the algorithms used can be found in [90].

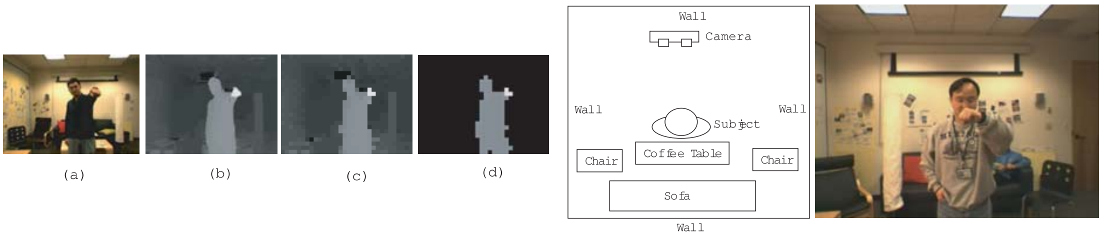


Figure 2.19: On the left: preprocessing of the depth images. (a) original frame; (b) depth map; (c) down-sampled map; (d) segmented map. On the right: test setup and a sample frame. (images from [90])

The gestures used in [90] to evaluate the gesture recognition method developed resemble boxing actions. A "defense" movement is crossing both fists in front of the head while "dodging" is to shift quickly aside. "Dashes" are straight arm-length punches thrown from the leading hand. A "hook" is swinging the arm, which is bent at nearly 90 degrees, into the opponent. Finally, a "uppercut" is a blow directed upward, as to boxing an opponents chin. Except "defense", all of these gestures can be performed with (or to) the left and right side [90]. These nine gestures were recorded from eight persons with different heights, weights and skin colors [90]. The recording setup and a person performing a boxing motion can be seen on the right side in Figure 2.19. By applying a leave-one-out cross validation the 336 recorded samples were used to test the gesture recognition. The evaluation led to an overall accuracy of 91.96% [90]. The detailed results of the evaluation for all boxing actions can be found in Table 2.2.

Action Name	Test Trials	Recognized	Accuracy [%]
Defese	40	39	97.50
Left Doge	32	30	93.75
Right Doge	40	38	95.00
Left Dash	40	35	87.50
Right Dash	40	36	90.00
Left Hook	40	35	87.50
Right Hook	16	15	93.75
Left Uppercut	40	34	85.00
Right Uppercut	48	47	97.92
Overall	336	309	91.96

Table 2.2: Results for the recognition of boxing actions from [90]

Overview on Related Work

After introducing five related works in detail this overview is intended to summarize these works. It is also intended to briefly present a few other related works. The works that were introduced above have been selected because they use a similar sensor and/or are intended to recognize similar gestures as are to be recognized in this study. The results from these works will be compared with the outcome of this study.

In [81] a framework for human activity modeling is introduced. After extracting reliable keypoints from a video clip, features are acquired by using the temporal self-similarities defined on the fuzzy log-polar histograms [81]. These features are used to train a SVM to realize an action recognition model. The method proposed in [81] is validated on two publicly available action datasets, the KTH dataset and the Weizmann action dataset [81] by using the leave-one-out cross validation technique. With this evaluation a recognition accuracy of 98.7% was achieved.
In [93] a set of time-sequential images is transformed into an image feature vector sequence. This sequence is converted into a symbol sequence by vector quantization. The symbol sequence is used as feature for the training and verification of HMMs. Experimental results in [93] led to a recognition accuracy around 90%.
The recognition of Malaysian sign language in video data is the topic of [37]. After the resolution of video data is reduced a skin segmentation takes place. After the skin segmentation has been undergone, all that is left in the images are blobs of hands and a blob of the head. Informations about these blobs (“Centroids”, “Distance between Hands and Face” and the “Hand Orientation”) are used as features to train HMMs. In [37] the HMMs are used for classification. To train the HMM 560 video sequences are used and another 112 video sequences are used for testing. Thus, recognition rates about 83% is achieved [37].
A method for recognition of pointing gestures is presented in [32]. This video-based system is intended to identify the points on a screen which a user is pointing to with his arm being in a fully extended position towards the screen [32]. The silhouette of the user was extracted from the video data. The information on the silhouette was used to extract features like the position where the person is standing, the position of the fingertip, and the position of the shoulder and to construct a feature vector for each video frame. This features are used to train a SVM in order to obtain the 2D position of the target point on the screen [32]. By using a leave-one-out cross validation and using four videos from two different persons a recognition accuracy of 94.4% was achieved.
In [29] a technique for a view-invariant detection of basic human actions like walking, jogging, hand waving and boxing is introduced. Sub-classifiers that are based on SVM are used to detect human body parts [29]. For each detected body the histogram of oriented gradients is used as a feature for the HMM that is used to classify human actions. For validation, the KTH database and the HERMES indoor sequence data set were used [29]. For the recognition of human actions an average accuracy of 79.5% was achieved.
To have a teleprocessing robot which is recognizing and following to military hand signals is the target of [16]. Military signals as “Halt”, “Crawl Forward” and “Retreat” should be recognized by the robot. In [16] the Kinect sensor is used to record human actions. Relevant features such as frequency and amplitude are used in a SVM to classify the gestures. By using randomized

offline testing recognition results with over 96% accuracy were achieved [16].
In [76] an engine to recognize dynamic hand gestures is introduced. A combination of static shape recognition, Kalman filter based hand tracking and a HMM based temporal characterization scheme is used as recognition strategy [76]. In an experiment five users were performing a total of 100 gestures [76] where 97% of the gestures were classified correctly.

The results of the related work show, that many different approaches for gesture recognition exist. Most of the results, even when using different methods, are leading to recognition results greater than 90%. The results for recognizing human motions do not only depend on the classification method used, but also on the selected features. The data used for evaluation has an influence on the evaluation results, too. Related work on gesture recognition using a similar sensor led to promising results when using the HMM for classification [42] [67]. A literature research conducted in [42] also showed that the HMM leads to promising results in human motion recognition. Table 2.3 gives an overview on the results of the related works.

Paper/Work	**Topic**	**sensor**	**recognition method**	**average recogntion rate**
Sadek 2011 [81]	Recognition of human actions	mono camera	SVM	98.70%
Yamamoto 1992 [93]	Recognition of tennis strokes	mono camera	HMM	>90.00%
Falinie 2012 [37]	Recognition of malaysian sign language	mono camera	HMM	83.00%
Cĕrnekova 2007 [32]	Recognition of pointing gestures	mono camera	SVM	94.40%
Chakraborty 2008 [29]	Recognition of human actions	mono camera	HMM	79.50%
Bernstein 2011 [16]	Recognition of military gestures	Kinect	SVM	9̃6.00%
Kristensson 2012 [53]	Recognition of dynamic hand gestures	Kinect	template matching	92.70%
Lang 2011 [56]	Recognition of german sign language	Kinect	HMM	97.70%
Ramamoorthya 2003 [76]	Recognition of dynamic hand gestures	mono camera	HMM	97.00%
Wang 2008 [90]	Using human body gestures as inputs for gaming	stereo camera	HMM	91.96%
Hahn 2011 [42]	Recognition of dance motions	UCOS family	HMM	97.33%

Table 2.3: Results of related works on gesture recognition

2.5 Summary of Related Work and State of the Art

In this chapter, related work and the state of the art for gesture recognition was presented. First, in Section 2.2, the topic of human computer interaction was examined leading to the conclusion that knowledge from various scientific fields is necessary to implement good user interfaces and good input methods. HCI is important for this work because the outcome of this study is a first step for the development of a system to control a multimedia user interface.

Section 2.2 presented informations on the meaning of gestures for human communication and ways to categorize human gestures. Because of the fact that the UCOS sensor is only capable of detecting motion [13], the gestures used in this study are dynamic gestures. If using the taxonomy introduced in [47], the gestures used in this work can be sorted in to the category of semaphoric gestures. These gestures are primary intended to communicate symbols to a machine [47]. When using the taxonomy from [70] which is more specialized on hand/arm gestures concerning HCI, the gestures defined for this work can be seen as deictic gestures and symbols. [47]

Information about gesture recognition are provided in Section 2.3. In this section an overview about the topic of gesture recognition is provided. The typical processing steps and selected classification methods are presented in this section. These processing steps are basically valid for all approaches about gesture recognition [62]. An overview on classification methods for gesture recognition was presented, too. In this section different types of features were explained, too. The approach of [80], in which basic geometric quantities were used as features, is very promising for this work as the trajectories produced by the UCOS sensor are similar to trajectories of movements of a computer mouse. Thus, a similar approach is used in this work. As there are many solutions possible for gesture recognition, finding the optimal method is a challenging task. Therfore, a more detailed review on selected related works about the recognition of gestures and human motions was presented in Section 2.4. In these related works, different sensors, different classification methods and different features were used. In most of the related works that were examined for this study, average recognition accuracies of 90% and better were achieved. In this work HMMs are used for gesture recognition because of the promising results that were achieved when a HMM classification and a sensor similar to the UCOS sensor were used for human motion recognition [42] [67]. Additionally, a literature study on human motion recognition in [42] shows, that the HMM is one of the most promising machine-learning methods for the recognition of human motion.

CHAPTER 3

Methodology

This chapter describes the steps of the methodological process of this study. The first section (Section 3.1) describes the dynamic stereo vision sensor and the AER data representing three dimensional movements. It also describes the special modifications of the AER data for gesture recognition. The second section (Section 3.2) presents a simple concept for a gesture controlled GUI. Section 3.3 presents a set of gestures based on the GUI concept which was introduced in the previous section, and the technological aspects concerning the UCOS sensor. Section 3.4 presents the methods used to obtain AER data and describes the preprocessing steps used before feature calculation. A database for the recorded data is presented in Section 3.5. Additionally a tool for navigation through the database and for annotation of the recorded data is also introduced in this section. The features used for gesture recognition are described in Section 3.6. In Section 3.7 two machine learning concepts for gesture recognition are introduced: the HMM and the DT. A trained decision tree can be interpreted as set of rules. It is seen as a “quasi-rule based” recognition method. In Section 3.8 the method for the evaluation of the gesture recognition is explained.

3.1 Dynamic Stereo Vision Sensor for Gesture Recognition

This section introduces the UCOS sensor and gives a short overview of its intended purpose and its technical specifications. In addition the AE data format produced by the sensor and the modifications necessary on the for gesture recognition are introduced here.

Intended Purpose

The UCOS stereo vision sensor developed by AIT is an integrated optical 3D sensing device that was originally developed for people counting applications (see Figure 3.2) [13]. In the monitoring area of the sensor persons are detected and counted according to their moving direction [13]. The detection of persons is based on the detection of movements at a certain distance from the sensor. Depending on the configuration of the detection zone, only movements in a certain distance are relevant for counting persons, making it possible to count only grown ups without children [13]. Figure 3.1 shows the typical application of this sensor. On the left side, the schematic of a passageway is displayed. When people move below the sensor, their heads are tracked. If a person passes a certain line underneath the sensor an increase of the count (of the direction the person is moving to) is triggered [13]. The visualization of the AER data generated when a person is moving below the sensor can be seen on the right side of Figure 3.1. For this work especially the visualization of the persons' moving direction by a trajectory is important.

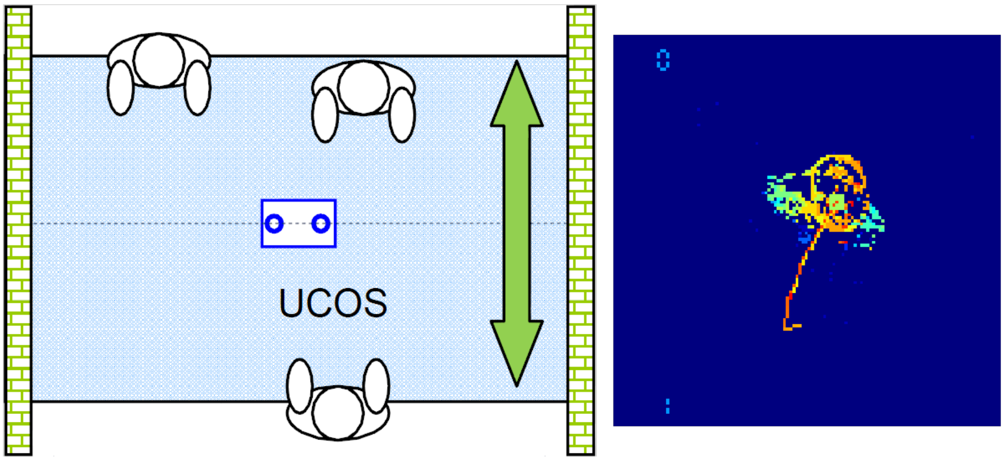


Figure 3.1: Current application for UCOS: counting persons (figure adapted from [13])

Technical Details

One can see in Figure 3.2 that the UCOS sensor system employs two specialized optical CMOS[1] sensors of 128x128 pixels which cover a photosensitive surface of 5.1 x 5.1mm². The distance between the sensitive pixels is 40µm [13] [59]. Each of the sensor's pixels quantisizes local relative light intensity changes to generate spike events [59]. This means that only the pixels which detect an intensity change produce an output [59]. This functionality is biologically inspired, therefore such an optical CMOS sensor is often referred to as a *silicon retina* [61]. At the output of the sensor these events appear as an asynchronous stream of data, the so-called address events (AE, TAE[2]), which code the pixel coordinates and the time of the intensity change in a stream of asynchronous vector information rather than in image frames [13] [59]. Further processing steps are used to multiplex the AEs of the two sensing elements to one AE-stream. A built in BlackFin digital signal processor from AnalogDevices with a CPU[3] core clock frequency of 600MHz, 32MB of RAM[4] and 4MB of non volatile memory is used to perform a stereo matching of the left and right image by using the SAD[5] Algorithm [9] [15]. The 3D depth information is also encoded in the AE stream which is available at the sensor's Ethernet LAN[6] connector in an UDP[7] data format [13].

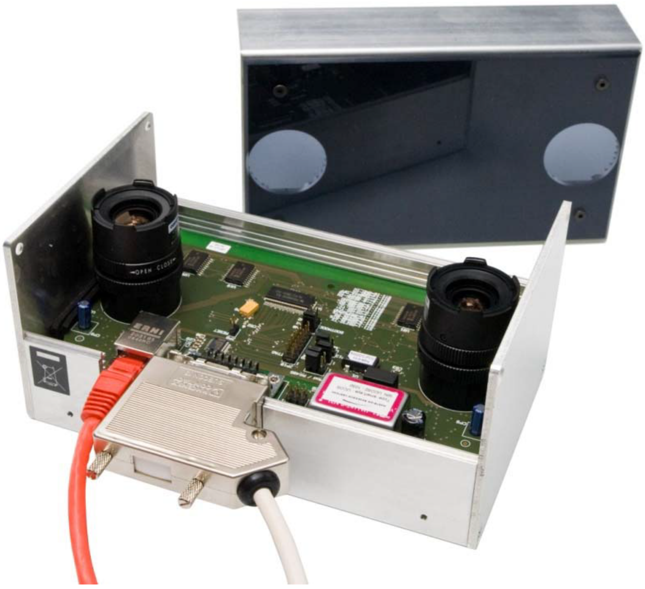


Figure 3.2: AITs UCOS Sensor (from [13])

---

[1] Complementary Metal Oxide Semiconductor

[2] Timed Address Event

[3] Central Processing Unit

[4] Random Access Memory

[5] Sum of absolute differences

[6] Local Area Network

[7] User Datagram Protocol

Generally, the sensor's principle of operation allows a robust and fast tracking of moving objects at up to 200 acquisitions per second [15]. Moving objects within a range of about three meters can be detected by the sensor [13]. Due to the fact that only movement is detected, no effort has to be put into separating a person from the background. The work of extracting the person is already done by the sensor [13]. Another advantage of the UCOS sensor is that it works in a wide dynamic range which allows operation also at bad light conditions (operation range 0.1Lux to 100kLux) [13]. This, combined with the other advantages mentioned above, suggests that the UCOS sensor is predestinated for indoor (and outdoor) motion recognition [13]. Examples for the sensor's output can be seen in the right part of Figure 3.1 and in Figure 3.3.

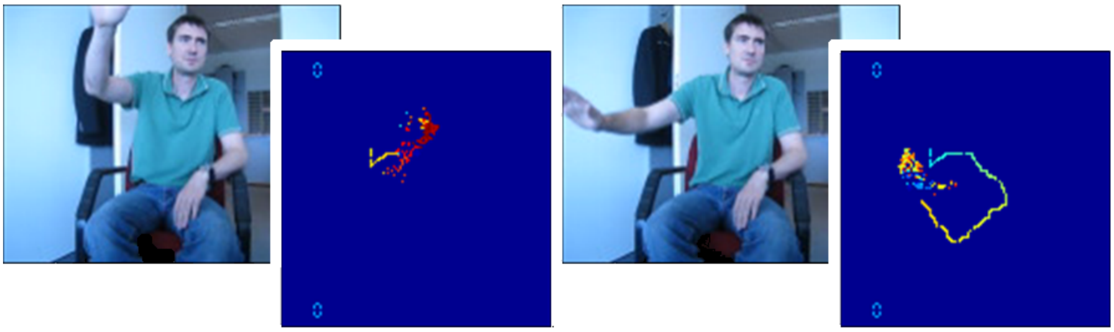


Figure 3.3: Person performing a circle movement in front of the sensor

Tracking

As mentioned above, the tracking of the highest point of persons (or objects) moving below the sensor is already implemented in the firmware of UCOS. When applied to count people, this feature is used to track the heads of people passing underneath the sensor (as illustrated in Figure 3.1) [13]. The tracks are visible for a longer time within the AER data than the AER data that represent the shape of a person. The higher persistence of the tracks is achieved by sending them not only once, but five times. The tracks are intended to visualize the movement of a person within the sensor data [13]. Not only the tracks are encoded within the AER data stream produced by the sensor, but also the numbers visible in the left upper and left lower corners on the right side in Figure 3.1.

If the sensor is not mounted overhead, but faces the person performing a gesture - as illustrated in Figure 3.4, tracking of the highest point becomes tracking the nearest body part which is in most of the cases a hand or an arm. The sensor detects the moving object, but does not make any differentiation whether the object is a hand, a head, a foot or even a pet running in front of the sensor. The output of the scenario when a person in front of the sensor performs a circular movement with their arm can be seen in Figure 3.3. To achieve optimal tracking results, the focus of the sensor should be in a concrete gesturing area in a predefined distance between one and three meters [13].

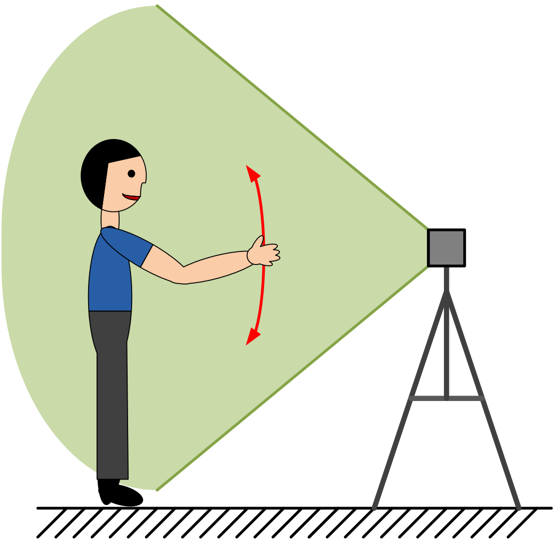


Figure 3.4: Hand/arm moved in front of the sensor

Special AER Format

As described earlier, the UCOS sensor streams its data, the so-called AER data, using UDP [13] [83]. If the option to visualize the the track within the AER data is activated, the numbers and the scan lines are encoded within the AER-Stream, too. Normally, there is no way to distinguish the tracks, scan lines and numbers from the other AER data that contains information on pixel activity - the actual information oft the object moving in front of the sensor [83]. For this work it is required to use the sensor's embedded tracking function. Therefore it is necessary to identify which type of AER data is actually received from the UCOS device. Thus, some small modifications of the original data format are necessary to distinguish useful data that contains tracks from unwanted data containing information which is not relevant for the current application. A comparison of the two data formats mentioned is illustrated in Figure 3.5. The original UCOS stereo data format is displayed at the top of the illustration and the modified data format is located in the lower part of the picture [83]. Two messages are transmitted for each pixel activity. These messages stay basically the same in the original and in the modified data format. Each of these messages has a length of 32bit. The messages are continuously transmitted alternating one after another. Normally the sequence of transmitted data packages should look like this: [TS AE TS AE TS ... AE TS AE ...] [42]. TS[8] is the time stamp of the activity while AE represents the location of the activity. It contains the Address (ADDR[9]), information on depth and other flags and messages [83]. The main difference between the unmodified and the more specialized data format is located in the little part, which contains the further information. Normally in the AE part of the message sixteen bits are used to encode the pixel activity address information while another eight bits are used to encode the depth information if existing. One bit

---

[8] Time Stamp

[9] Address

is used as a flag to signalize the existence and validy of the depth information. If this flag is not set, the bits containing the depth information are meaningless. The remaining seven bits out of the 32bit data package usually remain unused [83]. For the task-specific data format for gesture recognition another three bits get an important role. These three bits are used to distinguish between the different meanings an AER can have. In order to keep backward compatibility with the previously used AER format a setting of 0x00 represents the stereo events from the sensor. If the message bits are set to 0x01 the AER represents a pixel of the scan lines which can be encoded within the data stream. If set to 0x02 the received pixel information becomes part of the numbers that display actual counting value of persons that have passed the scan line. The most interesting setting for the gesture recognition is 0x03. There the received AER message contains a pixel which is part of the track that can be used at further steps in the recognition process. Finally, the message can contain the value 0x04 where the AER is part of a cross which can used to mark the beginning of a track [83]. For this work the setting 0x03 is the most important, because the output of the tracking function is used for further steps in the gesture recognition process. All other types of data have to be filtered out.

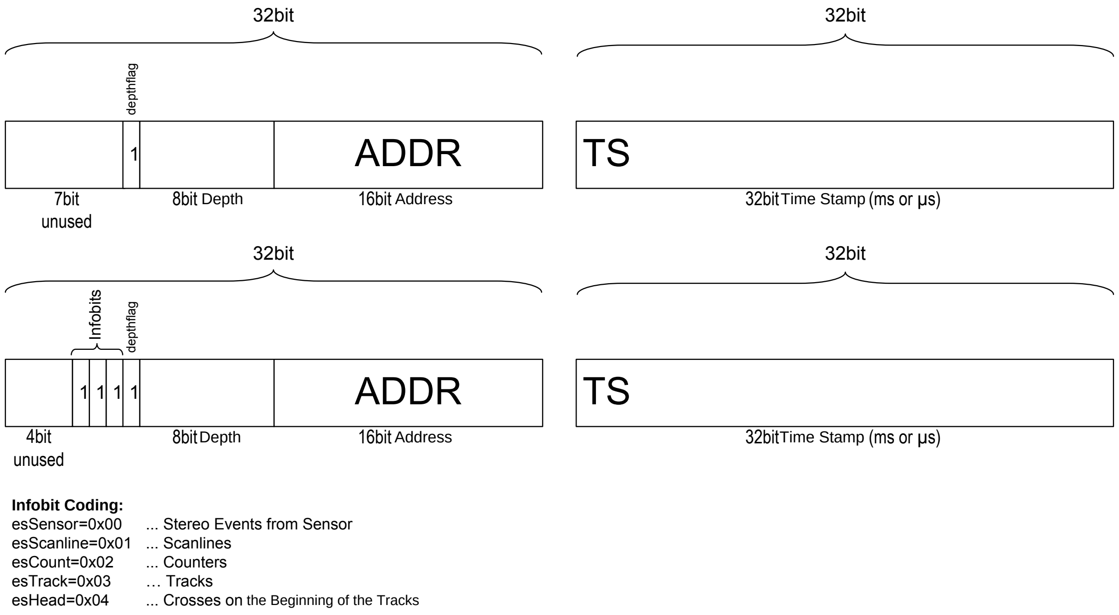

32bit
32bit
depthflag
1
ADDR
TS
7bit unused
8bit Depth
16bit Address
32bit Time Stamp (ms or µs)
32bit
32bit
Infobits
depthflag
1 1 1 1
ADDR
TS
4bit unused
8bit Depth
16bit Address
32bit Time Stamp (ms or µs)
Infobit Coding:
esSensor=0x00 ... Stereo Events from Sensor
esScanline=0x01 ... Scanlines
esCount=0x02 ... Counters
esTrack=0x03 ... Tracks
esHead=0x04 ... Crosses on the Beginning of the Tracks

Figure 3.5: Conventional UCOS AER format (top) and special AER format for gesture recognition (from [83])

3.2 Simple GUI Concept for Gesture Recognition

The design or the implementation of a GUI is not directly related to this work's task. However, to have an idea of what the gesture navigation could be used for, some thoughts have been spent on this topic. The better a system is designed for a selected input method the better it can be operated and the better is the user experience when working with it [38]. Therefore, to reach a coherent and usable system to be navigated by gestures, it is important that the applications - especially their GUI - is designed and optimized for gesture navigation. The gestures to be used for navigation have to correspond with the reactions of the GUI and vice versa [38]. The gesture set used for this work is defined and described in detail in Section 3.3.

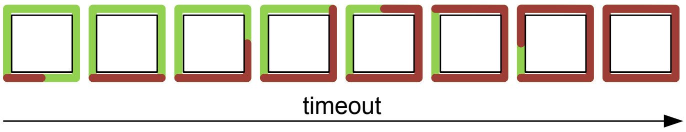


Figure 3.6: Visualization of the timeout before selection of a menu item

Generally, the proposed GUI has to be seen as a rudimentary concept of a multimedia interface for an application which uses a TV[10]-set as a display. It is one of the first concepts that have been introduced to the FoSIBLE project members. This GUI represents a basic concept that is intended to support the development of the gesture navigation system. The concept for the proposed GUI is quite simple. Basically, horizontal actions trigger a reaction of a horizontal menu and vertical actions trigger a reaction of a vertical menu. Therefore, the simplest set of gestures required to navigate this menus consists of four directional gestures. When the smallest possible gesture set is used, the selection of a menu item is to be triggered automatically by a timeout after the last movement of the user. As illustrated in 3.6, the timeout can be visualized by a sequential color change of the frame around the selected item.

In Figure 3.7 a screen displaying menus and an information region can be seen. The information region can contain multimedia data like text, pictures or video clips. In the information area no navigation takes place. Navigation is performed by moving the horizontal and vertical bars under the green selection window like a slot-machine (casino gambling machine) which turns the symbols in the players field of view. Figure 3.8 shows the same principle applied to a welcome screen that appears directly after activating the system (e.g. after turning on the TV on). In both GUIs, dark gray arrows are symbolize the direction of possible bar movements. This is intended to invite the user to perform gestures corresponding to the pointing directions of the arrows without much learning effort. Both screens have one thing in common: they can be navigated by using a reduced gesture set. A select or return gesture is not necessary for navigating through the menus.

---

[10] Television

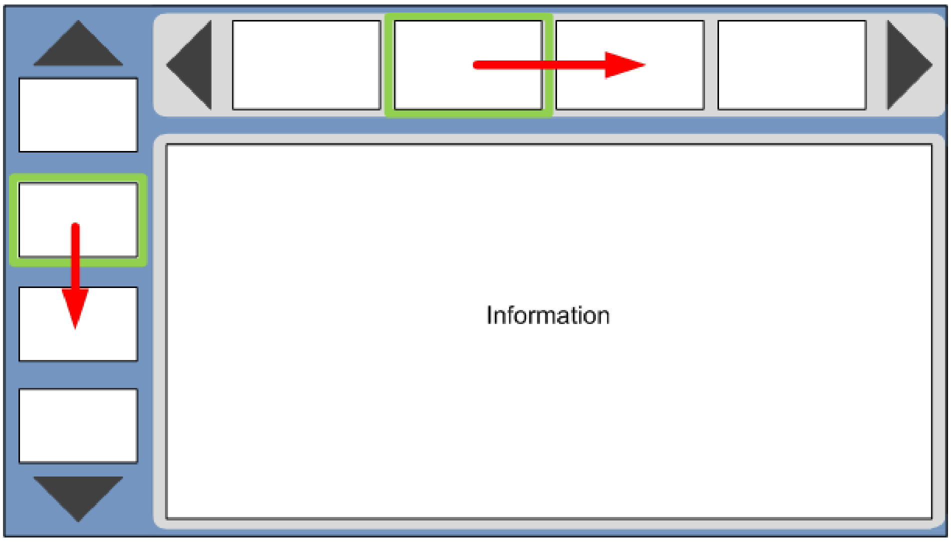


Figure 3.7: Screen containing information in a hypothetical GUI specialized for gesture control

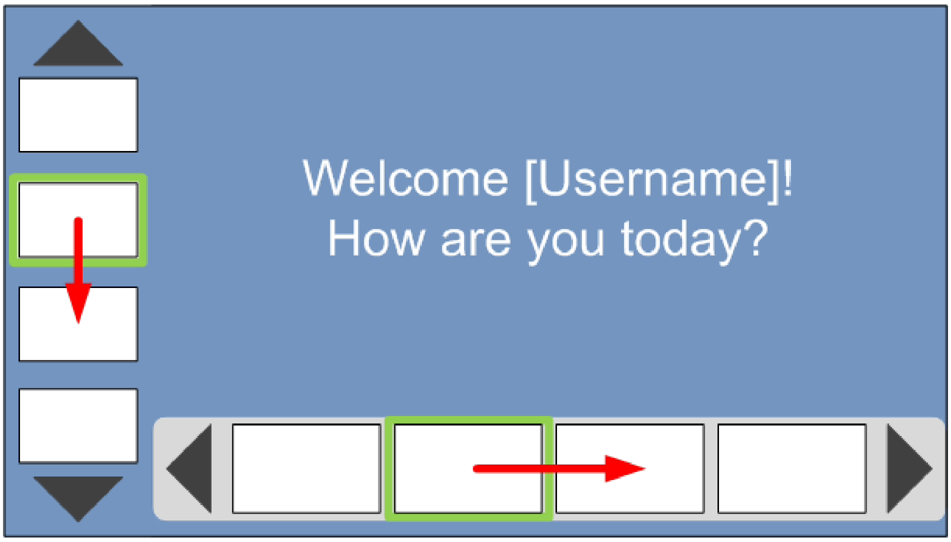


Figure 3.8: Welcome screen in a hypothetical GUI specialized for gesture control

Figure 3.9 shows a screen to be used to select pictures from a photo gallery. The typical view for selecting images from a photo gallery is a perspective where the thumbnails of the pictures are shown as tiles next to each other. Navigating through this field of items just in horizontal direction, jumping to the next line when reaching the end of the current line, is not very user friendly. To enable a navigation in all directions, it is necessary to deactivate the side menu, otherwise it would not be clear whether the gesture corresponds to the thumbnails or to the side menu. This is why the menu on the left side is grayed out in 3.9. Now the plane containing the thumbnails can be moved using horizontal and vertical gestures. Th plane moves below the selection window (symbolized by a green square). This can be compared with moving an object under a fix-mounted magnifying glass. A picture can be selected by stopping the movement when reaching it. If no further movement takes place the picture will be shown on the full screen after the timeout.

Until now navigation through the GUI is possible by using only four gestures. However, this is not very comfortable. There is also the question, how to leave the full-screen view or the thumbnail screen to return to previous menus. One answer is that the selection can be triggered by a timeout after the movement and “return” buttons can be implemented in the thumbnail view. The full-screen view of a picture can be reduced to a thumbnail after another gesture is detected. Another answer is to give more options to the user by introducing additional directions for the navigation to enable the users to move to all eight neighbors of a thumbnail picture. Four additional gestures in diagonal directions allow the plane that contains the thumbnails to be moved in eight directions. Giving the user the option of selecting an item by using a special gesture and to return to a previous menu screen by using another special gesture makes it a total of ten gestures to navigate this system. The gestures introduced in Section 3.3 can be used to navigate such a GUI.

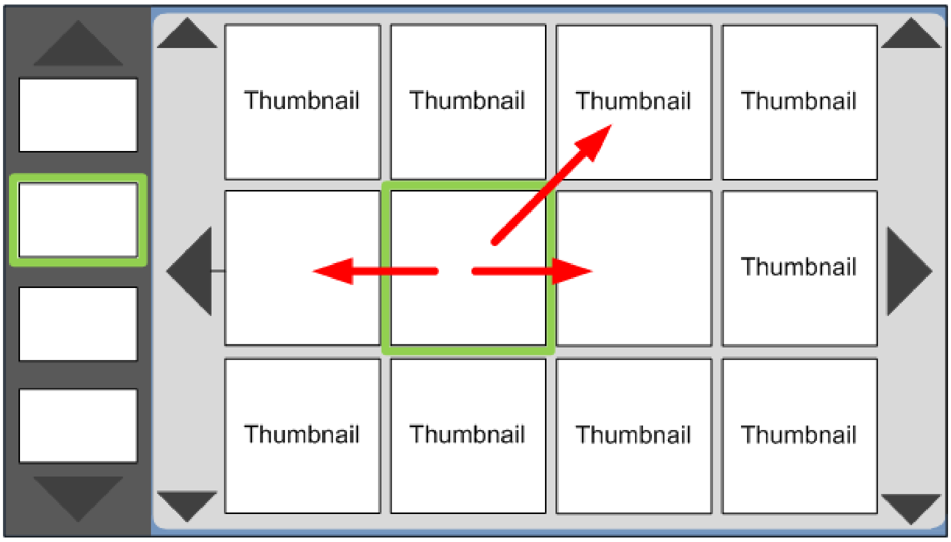


Figure 3.9: Screen containing thumbnail pictures in a hypothetical GUI specialized for gesture control

3.3 Defintion of Specific Gestures

Since humans can produce nearly a infinite number of different movements with their hands and arms [57], it was decided to define a specific and finite set of gestures for the implementation and the test process of a gesture recognition system. All gestures chosen for verification and testing of the gesture recognition based on AER data were defined with the intention to be used for navigation trough a simple multimedia menu as introduced in Section 3.2. Since the UCOS sensor used to generate the AER data only detects movement, only dynamic gestures can be used in this work [13]. A limited set of gestures, of which each gesture itself is kept simple, is used to keep the work load of learning the gestures for users as low as possible. For everyday use unnatural movements can not be used as gesture representatives [38] [20]. A set of ten relatively simple dynamic gestures was chosen as gesture representatives used for navigation - eight directional gestures and two gestures with non-directional and therefore special meanings. One of the special gestures is to be used to select an item and the other is used to return to a previous menu. In theory this predefined gestures have very distinctive directions and shapes, which makes it easy for humans to recognize the differences in the movements. Basically, this should also enable the differentiation of the meanings of the gestures for an artificial recognition mechanism, which is to be evaluated within this work.

Gesture name	Acronym
down	DO
up	UP
left	LE
right	RI
rightup	RU
rightdown	RD
leftup	LU
leftdown	LD
roof	RF
wave	WV

Table 3.1: Gesture names and short gesture names

Directional Gestures

The directional gestures defined for this work can be compared to the arrow keys on a computer keyboard. Normally, navigating through a menu by using the arrow keys means moving a cursor or the menu items into a certain direction. Therefore, directional gestures are considered a basic requirement for navigation tasks. These gestures are easy to understand and do not require much effort to be performed. A straight and distinctive hand/arm movement into a certain direction is interpreted as a directional gesture into the same direction. Moving the hand/arm in a horizontal axis means left or right, moving the hand/arm in a vertical axis means up or down. Navigation towards the corners of the screen is possible, too. To navigate into a diagonal direction the movements between the horizontal and the vertical axis are interpreted as diagonal gestures. A schematic of the eight directional gestures can be seen in Figure 3.10: there a person is performing the gestures using the right hand. The gestures can also be performed using the left hand. The amplitude of the gesture performed should be between 20cm to 60cm. The movement should be carried out using a deliberate movement. The directional gestures are named according to the direction of the movement: “down”, “up”, “left”, “right”, “rightup”, “rightdown”, “leftup” and “leftdown”. In this work acronyms are used that correspond to this names. These acronyms can be found in Table 3.1.

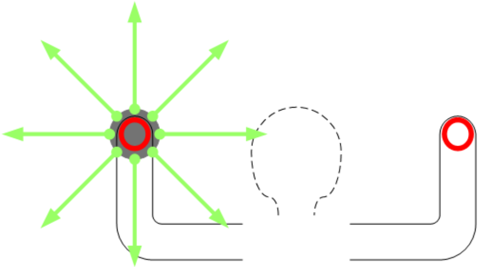


Figure 3.10: Eight directional gestures

Select Gesture

Navigating between menu items without the option of selecting any items is not intuitive [20]. Therefore, a commonly known and intuitive movement is intended to represent the user’s wish to select an item. The movement chosen for this task is waving the hand from left to right and left again. It’s like implying a “Hello!” by waving the hand. How often the hand is moved from one side to the other is not defined for this gesture, waving should be carried out until it is recognized by the system. A schematic of the “select” gesture is illustrated in Figure 3.11: there a person is waving with the right hand, but the gesture can also be produced using the left hand, too. The amplitude of the gesture should be between 10cm to 30cm. As for the directional gestures the movement should be carried out using a deliberate movement. The “select” gesture is named “wave”. It’s corresponding acronym can be found in Table 3.1.

Return Gesture

In the context of multimedia menu navigation the option of returning to the previous menu is necessary. Therefore, an additional simple gesture was defined: moving the hand/arm in a roof shaped pattern. Thus, moving the hand/arm diagonally up and diagonally down, without stopping the movement is to be recognized as the “return” gesture. A schematic of this gesture can be seen in Figure 3.12: there a person is performing a roof-shaped movement with the right hand. For the “return” gesture the overall movement of the hand/arm has to be to the right. This means, if the right hand is used to perform the gesture the direction of the movement has to be away from the body. If the left hand/arm is used to perform the gesture movement towards the body is required to keep the overall movement to the right. However, this limitation of this gesture is changed in a future state of the gesture recognition system. The amplitude of the gesture should be between 30cm and 60cm. As for all other gestures defined above, the movement should be done deliberately. Because it's roof-shaped pattern the "return" gesture is named "roof". It's corresponding acronym can be found in Table 3.1, too.

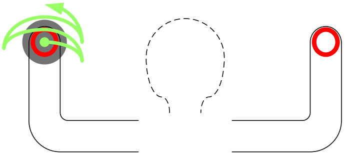


Figure 3.11: Hand waving as selection gesture

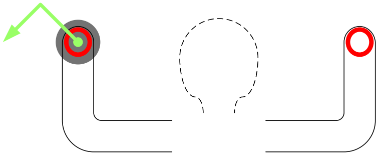


Figure 3.12: Roof shaped movement as return gesture

3.4 Data Recording and Preprocessing

Since gesture recognition is a new field of application for the AER produced by the UCOS sensor, no sample data is available so far. The test- and training data had to be recorded and preprocessed. This section describes the recording setup an the preprocessing steps required for this study. This section also gives an overview on how to extract tracks and how to translate these tracks to a format which can be used for further processing.

Recording Setup

For the FoSIBLE project the gesture recognition is applied in a living room environment [4]. A laboratory setting similar to the future every-day use scenario was built up for recording test and learning data for the evaluation process. The sensor and the person performing the gesture are placed in a similar distance and position as they are found in a normal living room environment. Figure 3.13 shows a schematic of a normal living room with the sensor placed on top of a TV-set, so a person standing in the gesture zone in front of the television can use his/her hand/arm movement to control applications on the TV-screen. As illustrated in Figure 3.4 the sensor can also be placed on a tripod instead of placing it on top of TV-set. Against the background that in future every-day use the sensor used in a living room and that similar distances to such an environment should be kept, it is recommended for the in-laboratory recordings that the sensor is mounted in a height of approximately 1.5m. The person performing the gestures has to be positioned in a central position in front of the sensor in a distance between 1.5m and 2m. As the living-rooms in future applications will be different, a variation of the recorded data is desirable. Therefore, the exact height and distances are not so important for the recording of test data, it is sufficient that the distance are kept approximately. The lenses of the sensor have to be facing towards the gesture zone and the operator. The illumination is not modified for test data recording, normal office light is used. Changes of the illumination are relevant only for for future test data recording. The settings of the UCOS sensor used in this study for recording gestures can are listed in Appendix C. It is important that the firmware which enables the special AER format introduced in Section 3.1 is loaded into the UCOS sensor [83]. To activate the output of trajectories the command “diagnose 2” has to be sent to sensor via console that is provided in SmartEyeCenter, a software briefly introduced in the next section [13].

Recording Software

For recording and controlling the UCOS sensor the software SmartEyeCenter developed by AIT has to be used on a PC [13]. This software is delivered together with the sensor on a memory medium [13]. Figure 3.14 shows a screenshot of the GUI from SmartEyeCenter running on a standard Windows PC. This software enables the user to display the sensor data, to configure the UCOS sensor and it offers a text-based console meant as an advanced configuration tool for experts [13]. An important aspect for evaluating the UCOS sensor’s potential for gesture recognition is that this software also has a built-in recording feature. This makes it possible to collect reusable data samples for the development process. The recorded data is stored in binary format (.bin), that can be directly read into MATLAB for further off-line processing.

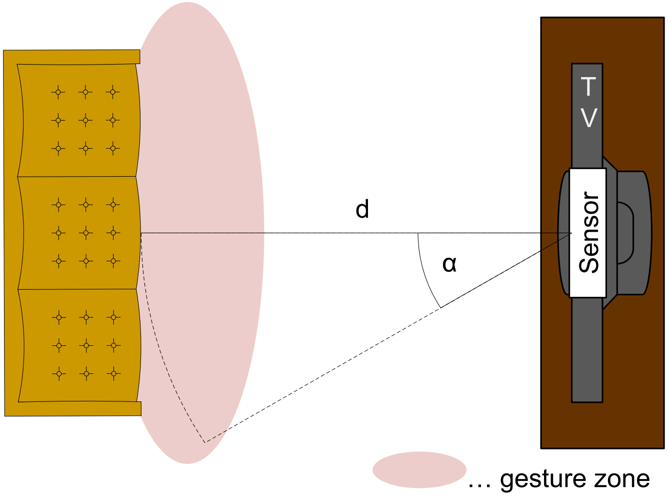
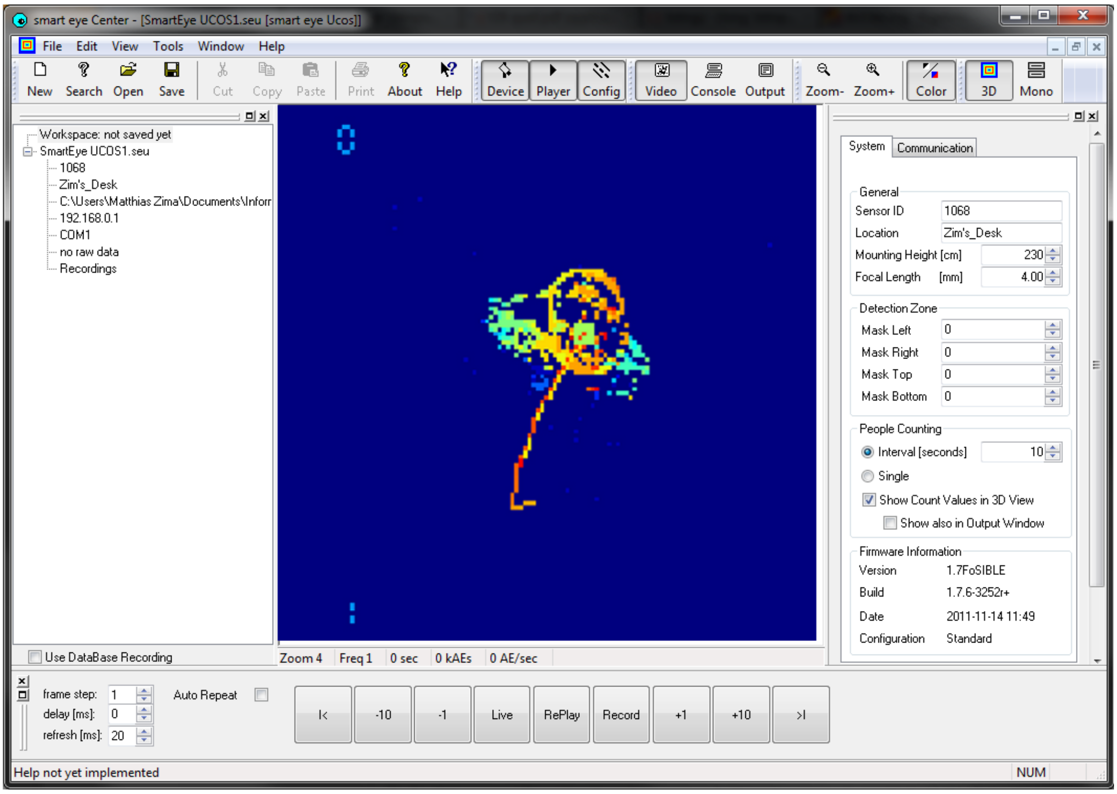


Figure 3.13: Setup for recording


File Edit View Tools Window Help
New Search Open Save Cut Copy Paste Print About Help Device Player Config Video Console Output Zoom- Zoom+ Color 3D Mono
System Communication
|< -10 -1 Live RePlay Record +1 +10 >|

smart eye Center - [SmartEye UCOS1.seu [smart eye Ucos]]

Workspace: not saved yet
SmartEye UCOS1.seu
- 1068
- Zim's_Desk
- C:\Users\Matthias Zima\Documents\Inforr
- 192.168.0.1
- COM1
- no raw data
- Recordings

☐ Use DataBase Recording

Zoom 4 | Freq 1 | 0 sec | 0 kAEs | 0 AE/sec

General
Sensor ID: 1068
Location: Zim's_Desk
Mounting Height [cm]: 230
Focal Length [mm]: 4.00

Detection Zone
Mask Left: 0
Mask Right: 0
Mask Top: 0
Mask Bottom: 0

People Counting
● Interval [seconds]: 10
○ Single
☑ Show Count Values in 3D View
☐ Show also in Output Window

Firmware Information
Version: 1.7FoSIBLE
Build: 1.7.6-3252r+
Date: 2011-11-14 11:49
Configuration: Standard

frame step: 1
delay [ms]: 0
refresh [ms]: 20
Auto Repeat ☐

Help not yet implemented | NUM

Figure 3.14: Application for recording and sensor control: SmartEyeCenter

Finding Tracks

At first the data has to be converted from binary to a format that can be further processed by MATLAB. Since the AER stream received from the sensor does not only contain track information [13], the relevant data has to be exported. This is done by applying a bit mask to the received data frames to remove all packets which do not match the bit mask. The result of this procedure leaves pure track data for further processing. Since the original use of the tracks is to visualize the direction a person underneath the sensor is walking, a higher persistence of the AER data containing the tracks is implemented by sending this data five times. Therefore, additional work has to be done if the tracks are used for gesture recognition, as the increased persistence is not useful for this application. These multiples of the tracks have to be removed, because only one track is used for feature calculation. Short tracks with less than ten points are also not useful for gesture recognition since they mostly represent unintentional small movements. Therefore, they are removed from the data, too. Figure 3.15 shows the difference between the initially received track data (left) and the reduced data (right), after the initial preprocessing.

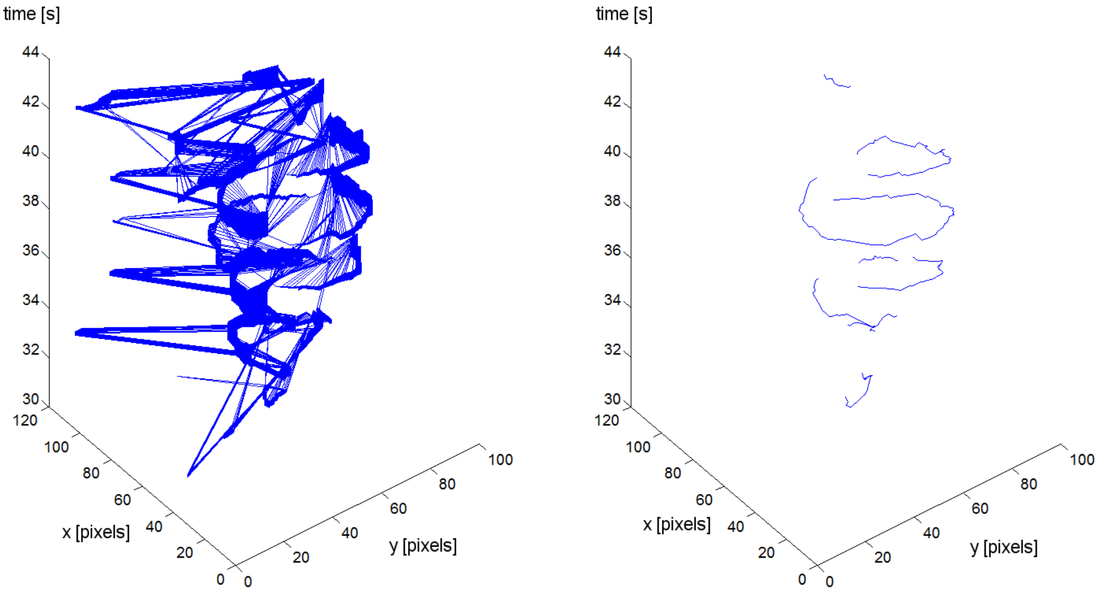


Figure 3.15: Raw and processed track data

Figure 3.16 shows a more detailed overview of the steps which have to be performed to extract relevant track data from the recorded data. When the binary file is loaded into MATLAB a chain of transformations of the data passes through. First, the binary information is converted into the now 32bit long words described in Section 3.1 by using the function `ae_bin2mat.m`. Then, the function `ae_trackfilter.m` applies the bit mask to the AE word so that all the information that does not contain track data is ignored during the further processing steps. For easier handling and better readability information like pixel coordinates, depth and time stamp are now extracted from the two 32bit long words by the function spverb|eth2itn.m|. This information is stored in a structure called *ITN Structure*. As described earlier, this data is used for the removal of doubles and tracks that too small for further use. The function `ae_trackfilter.m` is custom built for this study. A listing of relevant parts of the code can be found in Appendix B. The other two functions were provided from AIT.

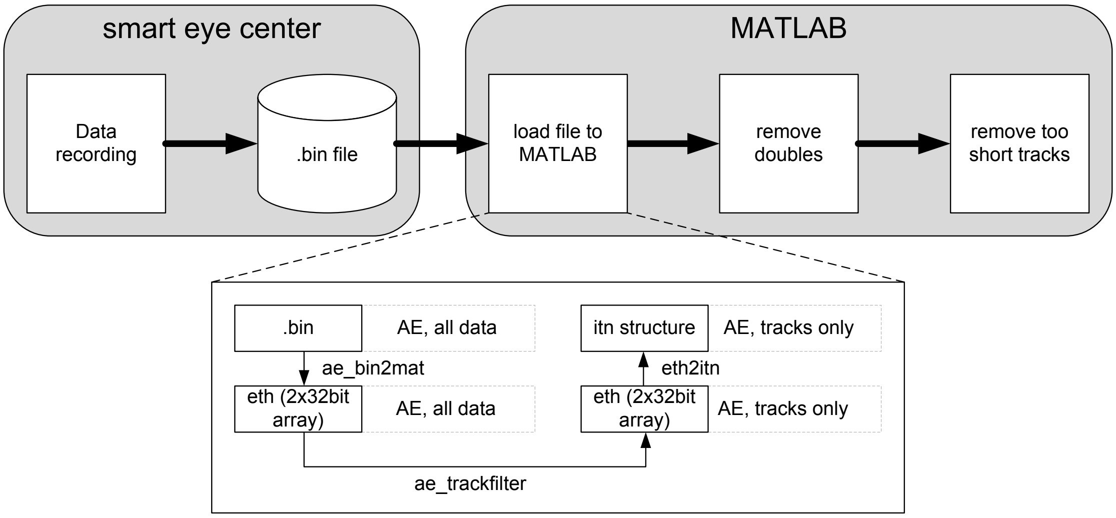


Figure 3.16: Steps from data recording to tracks ready for annotation

3.5 Test Database and Data Annotation

For further use the recorded test data has to be organized and sorted. Therefore, a database with a simple structure and a GUI called "TracksortGUI" for annotating and editing the data were developed. The data in this database is then used for the training and the verification of the gesture recognition mechanism. A brief description of the database and a short introduction into the GUIs functions is prsented in this section. Also the quality criteria for a track are discussed in this section.

Database Structure

Basically, the test and learn database consists of two MATLAB data structures. One structure is used to store information about files that were already processed. The other structure is used to store information about the tracks that were annotated. These tracks can be used for the training and the verification of the gesture recognition mechanism. For each file TracksortGUI generates an entry that contains: the file name, the directory and the name of the computer that was used to annotate the file. Unique ID[11]s of the tracks extracted from this file are stored in the file information. The structure containing the track data consist of an array of structures itself. For each detected and annotated track an entry is generated that contains the AE information (e.g. coordinates and time stamps), the recording parameters (e.g. sensor distance, person performing the gesture, light conditions) and the quality criteria that enables the selection of better or worse executed gestures. The type of the gesture represented by the track is stored in this entry. [71]

Anntotation Tool

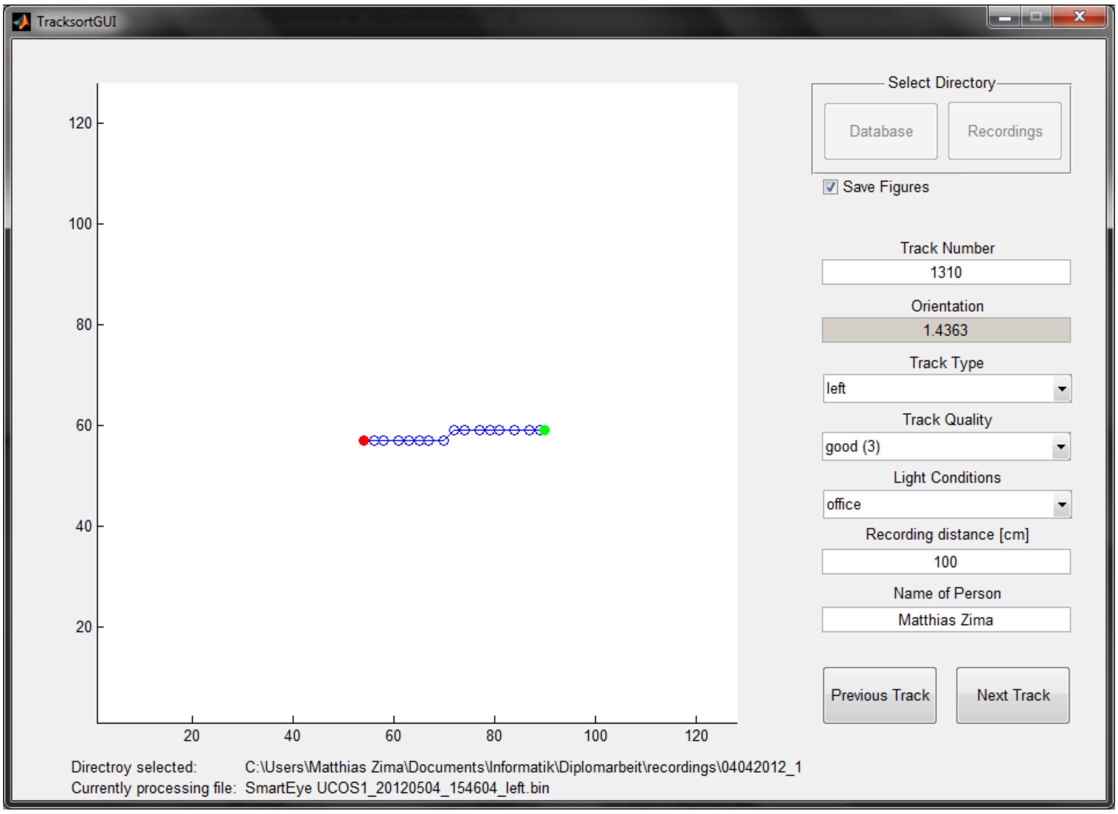
In order to annotate and sort the recorded track data it is necessary to visually inspect and annotate the recorded track data. This is done by using the tool TracksortGUI that was programmed in MATLAB for this study. It is specialized to work with the track data generated by the UCOS sensor. If a new collection of database should be crated, at first, a new directory to store the database has to be defined. Alternatively, an existing database can be edited or extended by using this tool [94].

To add new track data to the existing or empty database, a directory containing previously recorded files has to be selected. In the directory, all files that have not been annotated yet are opened in a row, one after another. Then the tracks are extracted from the file according to the process described in Section 3.4. Each track is displayed separately in TracksortGUI. The beginning of the displayed track is marked with a green dot, its ending is marked with a red dot. Now, the track can be inspected visually by an operator and sorted to a certain gesture class according to the definitions in Section 3.3. The quality of a track can be rated, too, by using this custom built tool. However, quality statements on the recorded trajectories are a subjective issue. A track that looks like the gestures described in Section 3.3 without deviations has to be tagged as "perfect (5)", a track with minimal deviations "very good (4)". Tracks that have deviations but still clearly resemble certain gestures have to be rated as "good (3)". If the deviations are bigger the rating has to be "acceptable (2)". If a trajectory is nearly unidentifiable for the human

---

[11] Identifier that uniquely identifies an object or record

eye it has to be tagged as *bad (1)*. Tracks which do not resemble any of the predefined gestures have to be considered as *invalid (0)*. A more detailed description of TracksortGUI can be found in [94].



☑ Save Figures

Track Number: 1310

Orientation: 1.4363

Track Type: left

Track Quality: good (3)

Light Conditions: office

Recording distance [cm]: 100

Name of Person: Matthias Zima

Directroy selected: C:\Users\Matthias Zima\Documents\Informatik\Diplomarbeit\recordings\04042012_1

Currently processing file: SmartEye UCOS1_20120504_154604_left.bin

Figure 3.17: GUI for track annotation

3.6 Features

All gesture recognition approaches that were reviewed for this study imply that distinguished features have to be calculated [67] [42] [87] [62] [56] [90] [76]. In this section the features used in this work are introduced. These features were kept simple, because of the future target to implement the gesture recognition system in an embedded device with limited computational resources. Therefore, features that are primary based on geometric information like angle and length - similar to those in [80] - are used in this work.

Length

The length of a track is defined as the accumulated sum of all distances between adjacent points. The distance between them is calculated using the Pythagorean Theorem [82]. This can be seen in Equation 3.1. The result of this calculation is then summarized to the previously calculated distances as can be seen in Equation 3.2.

$$\begin{aligned} \Delta x_0 = x_1 - x_0, \Delta x_1 = x_2 - x_1, ..., \Delta x_{n-1} = x_n - x_{n-1} \\ \Delta y_0 = y_1 - y_0, \Delta y_1 = y_2 - y_1, ..., \Delta y_{n-1} = y_n - y_{n-1} \end{aligned} \tag{3.1}$$

The variables Δx and Δy represent the difference of the coordinates the neighbor points in a Cartesian coordinate system. They can also be interpreted as an adjacent and opposite site of a right triangle which enables the use of the Pythagorean theorem [82] and basic trigonometric functions [82] for further calculations.

$$length_n = \sum_{1}^{n} \sqrt{\Delta x_n^2 + \Delta y_n^2} \tag{3.2}$$

Speed

Another possible feature is the speed of the expansion of a track. The parameter is calculated using the distance between two points and also the time difference Δt is used to calculate the speed. Again, the variables Δx and Δy represent the distance between the points in a Cartesian coordinate system. The calculation is done in the same way as for the length, using the Pythagorean theorem [82]. Since the speed is defined as the covered distance divided by time [43], the differential time stamp Δt between two activities is used for this calculation. This can be seen in Equation 3.3

$$speed_n = \sqrt{\frac{\Delta x_n^2 + \Delta y_n^2}{\Delta t_n}} \tag{3.3}$$

Orientation

Another feature that is applied for identification of the meaning of a track is the orientation. The orientation of a track can be defined in different ways. In this case, two different definitions are chosen. One option is to calculate the angle between adjacent points; the other option is to calculate the angle between the starting point of a track to each of the following points. The absolute orientation of a track can be seen as the average value of all orientations collected. This average value is also used used to make an approximation where the track is moving to.

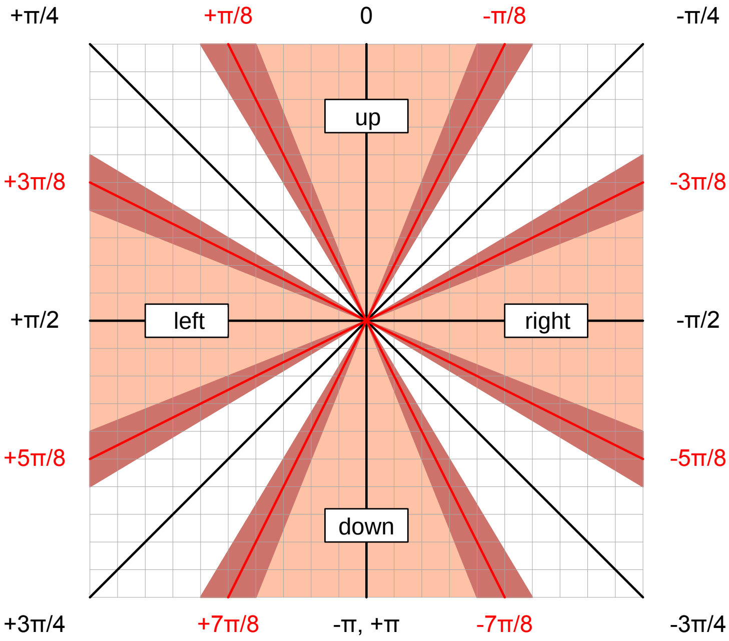


Figure 3.18: Meaning of angles

In the calculation of the length the variables Δx and Δy represent the difference of the coordinates of the points in a Cartesian coordinate system. Therefore, the calculation is done the same way as for the calculation of the length. These values are used to calculate the angle in the rectangular triangle formed between the adjacent points. To do so, a trigonometric function - the four-quadrant inverse tangent - is used [82]. The results of this function have a value between $-\pi$ and $+\pi$. The meaning of such an angle value of the data produced by the UCOS sensor is illustrated in Figure 3.18.

$$orientation_n = arctangent(\Delta x_n, \Delta y_n) \tag{3.4}$$

Orientation between neighbor points

The initial way to calculate the orientation is to calculate the orientation to the nearest neighbor point. This method is depicted in Figure 3.19. For more flexibility, it is also possible to calculate the orientation to a neighbor point in a certain distance of *n* points.

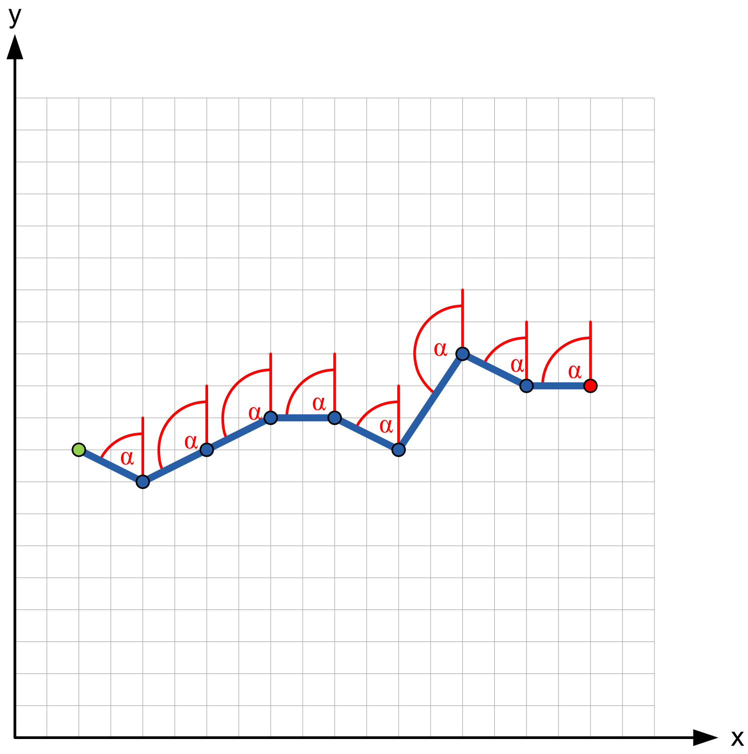


Figure 3.19: Calculating the orientation between neighbors

Orientation between first and current point

An alternative to the calculation of the orientation between neighbors is calculating the angles between the starting point and all other points of a used track as feature. Basically, the calculation of the angle is done in the same way as the calculation for the orientation between adjacent points. However, the variables Δx and Δy are now calculated in relation to the first point of the track [82]. This method is illustrated in Figure 3.20.

$$\begin{aligned} \Delta x_0 = x_1 - x_0, \Delta x_1 = x_2 - x_0, ..., \Delta x_{n-1} = x_n - x_0 \\ \Delta y_0 = y_1 - y_0, \Delta y_1 = y_2 - y_0, ..., \Delta y_{n-1} = y_n - y_0 \end{aligned} \tag{3.5}$$

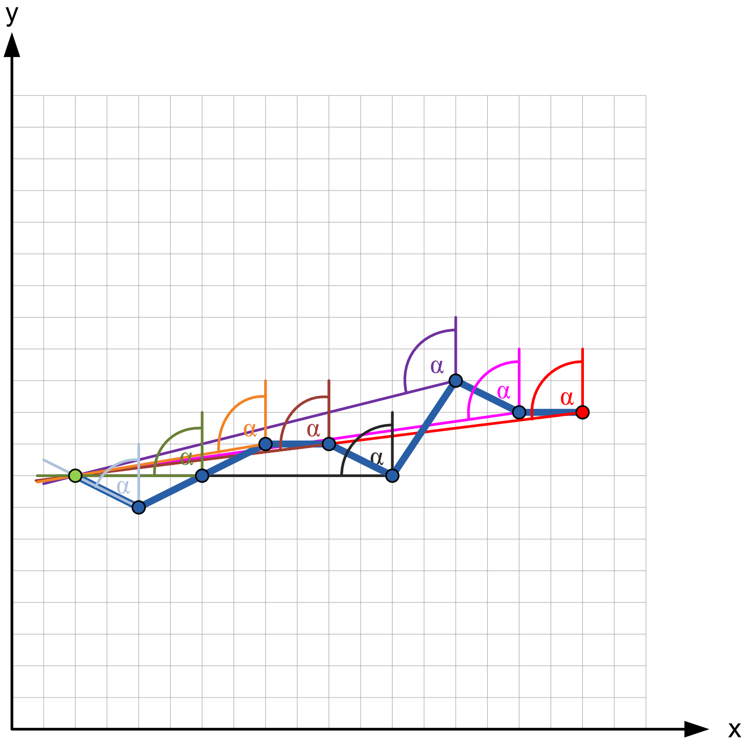


Figure 3.20: Orientation between first and following points

Bounding Box Area and Ratio

The Bounding Box (BB[12]) is the area covered by a box around the gesture movement [67]. The surface area and the ratio of the lengths of the sides of the box are used as features in this work. To calculate these characteristics, the maximum and minimum value for x and y have to be determined. This is done by comparing these values to each other. When a minimum or a maximum is found, the value is stored and saved as the new minimum or maximum value. Then this value is compared to the rest of the values. When the minimum values and the maximum values for x and y are determined, the bounding box characteristics are calculated. This is done everytime a coordinate is checked, thus, the bounding box characteristics change everytime the maximum and the minimum value change.

Since the Bounding Box is a rectangle (red in Figure 3.21), the covered surface area (light red in Figure 3.21) is calculated by multiplying the sides that form the rectangle with each other [82]. The length of the sides (Δx, Δy in Figure 3.21) is represented by the four most distant coordinates (xmax, xmin, ymax, ymin in Figure 3.21). The ratio between the sides is used as a feature, too. If the side Δy has a size of zero, the ratio between the sides is also set to zero. The

---

[12] Bounding Box

BB characteristics are calculated for each pair of variables while the vector containing the track information is stepped trough by a loop. If there is no change, the old value is kept as characteristic of the current trajectory. In case of a change, the BB-ratio is modified, whereas only the maximum of the BB-area is kept. The way to calculate the BB features is showed in Equation 3.6

$$\begin{aligned}
\Delta x_{n-1} &= max(x_{0...n}) - min(x_{0...n}) \\
\Delta y_{n-1} &= max(y_{0...n}) - min(y_{0...n}) \\
BBArea_{n-1} &= \Delta x_{n-1} * \Delta y_{n-1} \\
BBRatio_{n-1} &= \frac{\Delta x_{n-1}}{\Delta y_{n-1}} ...for \Delta y_{n-1} > 0 \\
BBRatio_{n} &= 0 ...for \Delta y_{n-1} = 0
\end{aligned} \tag{3.6}$$

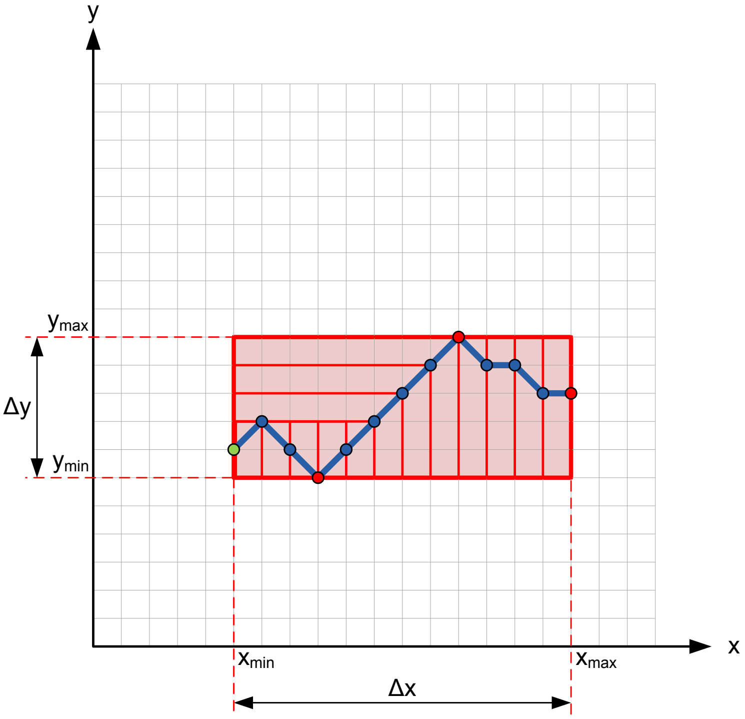


Figure 3.21: Schematic of a Bounding Box

Sensor Segmentation - Direction Changes

The directional gestures defined in Section 3.3 are moving to one direction. On contrary, waving and drawing a roof in the air as defined in Section 3.3, involve more direction changes. This leads to the conclusion, that the number of direction changes is another useful feature for gesture recognition based on AER trajectories. Thus, a simple approach was chosen to detect directional changes which is described here: the sensor surface, respectively the coordinates of active pixels, were segmented into horizontal and vertical zones. The number of zones is set in the range between one and the size of the sensor - 128 in case of the UCOS sensor. It is appropriate to use values which are in a range of $\frac{128}{2^n}$ to reduce rounding errors. As illustrated in Figure 3.22, the zones are numbered from 1 to n. When the track moves through the zones, the algorithm registers if the identifiers of the zone are increasing or decreasing. A change from increasing numbers to deceasing numbers is recognized as a direction change. This is done separately for horizontal and vertical movements.

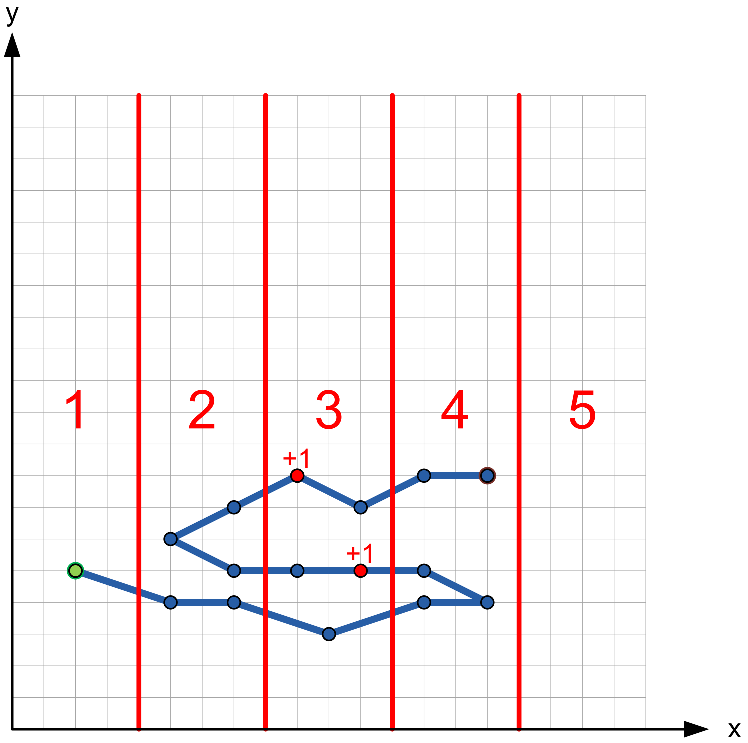


Figure 3.22: Sensor segmentation and direction changes

3.7 Machine Learning Based Gesture Recognition

Since related work on gesture recognition using a similar sensor led to promising results when using the HMM method for classification, the HMM method is used in this work, too. A literature research conducted in [42] also showed, that the HMM leads to promising results in human motion recognition. The second method evaluated in this study is the DT. The DT is selected because it generates a set of rules as output that can easily be used in future embedded implementations of a gesture recognition system based on the UCOS sensor. In this section, Hidden Markov Models (HMM) and the Decision Tree (DT) are introduced. It also describes how the HMM and the DT are trained and used for gesture recognition.

Introducing Hidden Markov Models

Formally speaking, a HMM is a variant of a finite state machine that contains two random processes. The first of them is the invisible so called Hidden Markov Chain [36]. The Hidden Markov Chain is a mathematical system that examines chain-like transitions from one state into another, under the condition that the current state depends only on the preceding state. The HMM is characterized by the states $q(t)$ and the transition probabilities a_{ij} between the states. The second process generates the visible output $o(t)$, also known as *observation symbols*, with probability distribution b_{ik} that depends on the states of the hidden process [67] [19]. This formal definition is depicted in Figure 3.23 which is adapted to fit to the example presented in [36] and [75]. In [36] and [75] the same description of the basic principals and notations for HMMs is given: consider a person that is tossing three coins in a closed room. It is not visible how the coins are tossed. Only the outcomes of the coin flips are shown on a display outside of the room. In other words, a series of hidden coin tossing experiments is performed. The only visible result is the output sequence consisting of tails and heads (e.g. TTTHHHTHTTHHT). This series of observations is called *observation sequence*. For people outside the room the sequence in which the different coins are is tossed is unknown, as is the probability of selecting the different coins. To estimate how much the outcome depends on the biasing of the coins and the order of the tosses, it may be presumed that tosses of the third coin result in more heads significantly, but all coins are thrown with equal probability. Naturally, it could be expected that with these prerequisite a far greater number of heads would be found in the output sequence. But if the bias probability of tossing the third coin (state) after either the first or the second coin (state) is zero and assuming that tossing begins in most of the cases with the first or the second coin (state) the outcome of heads and tails will be almost equally distributed, in spite of the bias. It is clearly visible that the output sequence strongly depends on the individual bias of the coins, the probabilities of the transition between the states and on the initial state to begin the observation. These three sets characterize what is called the HMM of this tossing-the-coin experiment.

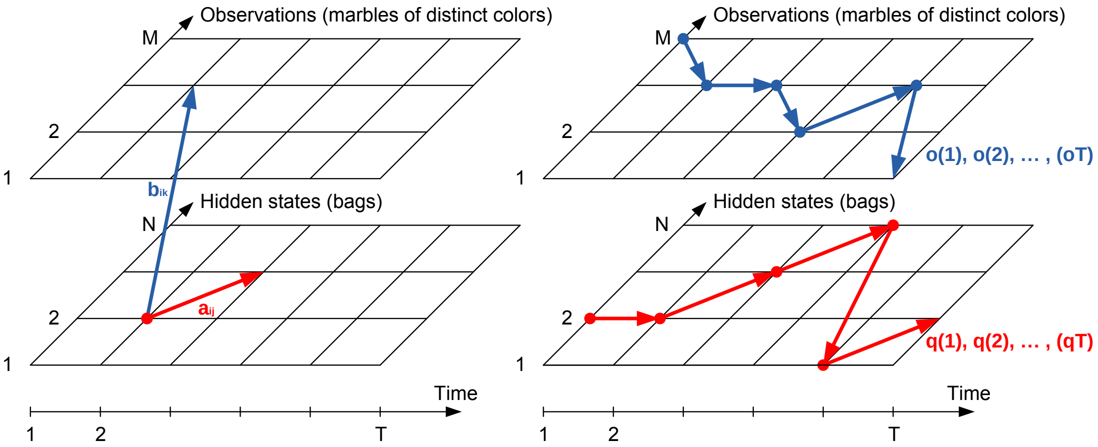


Figure 3.23: Hidden Markov Model (graphics adapted from [85])

Both [36] and [75] introduce the marble and urn model to extend the ideas of the HMM to a more complicated situation. As depicted in Figure 3.24, this model consists of N large glass urns placed in a room. Each of them is filled with a large number of colored marbles. The marbles in the urns have M different colors. The process of obtaining an observation can be described as follows: In the room full of urns a mastermind picks an initial urn according to a random process. From this urn a marble is picked, also at random. The color of the selected marble is noted as observation. The marble is then put back into the urn it was selected from and then a new urn is chosen according to the random selection process that is associated with the current urn. Then the marble selection process is repeated. As noted in [75], this entire process generates a finite observation sequence of colors, which could be modeled as the visible output of a HMM. The simplest HMM that corresponds to the marble and urn game is the one in which each state corresponds to a specific urn and for which the marble-color probability is defined for each urn (state). The state transition matrix of the HMM dictates the choice of the urns (states).

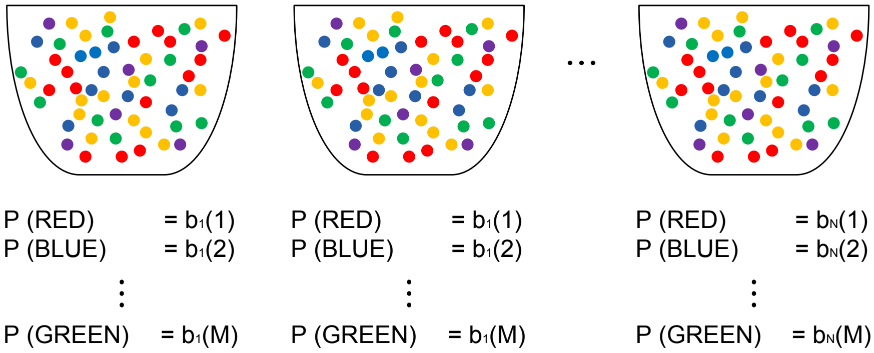


Figure 3.24: A N-state urn and marble model to illustrate the general case of a discrete symbol HMM (graphics adapted from [75])

Referring to the short formal definition and as illustrated by the marble example the characterizing parameters of the HMM can be summarized as follows: [75] [67] [36]

- **Hidden states**:
  $Q = \{q_i\}, i = 1, ..., N$
  N represents the number of states (bags in the marble example or biased coins in the coin-tossing example.)
- **Observations:** $O = \{o_k\}\, k = 1, ..., M$
  M represents the number of distinct observation symbols (marbles of different colors, or heads and tails in the example)
- **Transition probabilities:**
  $A = \{a_{ij}\}$ where $a_{ij} = P(q_j \ @\ t + 1|q_i \ @\ t)$
  $P(a|b)$ is the conditional probability of a given b. This means, a_{ij} is the probability that the state q_i is proceeded by the state q_j.
- **Emission or output probabilities:** $B = \{b_{ik} = b_i(o_k) = P(o_k|q_i)\}$
  $P(o_k|q_i)$ is the conditional probability that the output is the observation o_k, given that the current state is q_i.
- **Initial state probability:**
  $\pi = \{p_i = P(q_i \ @\ t = 1)\}$.
  π specifies the probability to be in state i at the beginning of the experiment, at time $t = 1$.

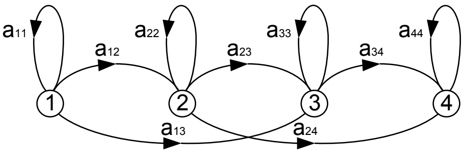
To denote a HMM, a so called triple $\lambda = (A, B, \pi)$ is used. The states Q and the outputs O are usually self evident [85].

In literature three main problems of HMMs which have been solved for most applications are pointed out: [36]

1. If a model $\lambda = (A, B, \pi)$ is given: how to compute $P(O|\Lambda)$, the probability of the occurrence of the observation sequence $O = O_1, O_2, ..., O_T$?
2. If a model $\lambda = (A, B, \pi)$ is given: how to find the most likely sequence of states for a given output sequence, so that $P(O, I|\Lambda)$ is maximized?
3. How to adjust the HMM parameters $\lambda = (A, B, \pi)$ in order to maximize $P(O|\Lambda)$ or $P(O, I|\Lambda)$?

These three problems can be solved by applying the following algorithms: The first problem, computing the probability of occurrence of an observation sequence, can be solved by using the *Forward and Backward* algorithm [75]. The second problem, finding the optimal state sequence, can be solved by using the *Viterbi* algorithm [75]. By applying the *Baum-Welch* algorithm, the third problem, finding the best matching state transition and output probability, can be solved [75]. More information about the employment of these algorithms can be found in [85], [75] and [36].

There is more than one type of HMMs [75]. A special case is the ergodic or fully connected HMM in which each state can be reached from every other state (including itself) in a finite number of steps [75]. For some applications - including gesture recognition - a HMM, in that not each state can be reached from all other, models better than the ergodic model [75]. Signals that change over time can be modeled better by using the left-right type of HMM, in which the states proceed from left to right [42]. Such a left-right model is depicted in Figure 3.25. It is visible, that in this model only transitions from a state to itself or a state more on the right are possible.



$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix}$$

Figure 3.25: A 4-state left-right model and its corresponding state transition matrix (graphics adapted from [75])

Training and Recognition with Hidden Markov Models

For the gesture recognition introduced in this study, the calculated features are used as observations to train the HMM. Not only one HMM, but one HMM for each gesture has to be trained [19]. For example a HMM that is meant to detect the gesture "right" is trained with features calculated from the corresponding recorded gesture. The same is done with the HMMs for the other ten gestures, presented in Section 3.3. As depicted in Figure 3.26, these ten individually trained HMMs work in parallel. To recognize a gesture, each of the HMMs is presented with the same feature set of an unrecognized track. Each of the HMMs will use these features to calculate whether the features of the unknown track fit to the model. The decision which gesture is represented by the analyzed feature set is made by selecting the maximum likelihood out of the results.

The features of a track are combined to an array. In this array a feature vector for each point in time is formed. This feature vectors are attached to each other in the sequence they occur to construct a feature cell array. By doing so, an array of feature vectors is constructed for each track. For training the feature arrays for all tracks representing a specific gesture are combined to a cell array. This cell array represents the training data for one specific gesture, thus it is used to train the HMM that intended to recognize this gesture. The process of generating the cell array is depicted in Figure 3.27. As mentioned earlier, each gesture is to be recognized by a custom trained HMM [19]. Therefore, the process of generating a feature cell has to be repeated for each gesture. The Toolbox from [64] with modifications from [78] is used to train the HMMs and to use them for classification.

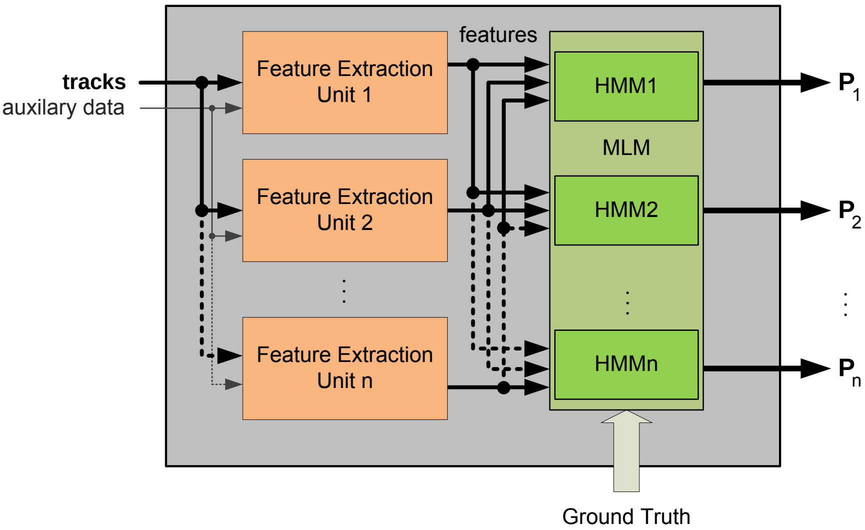

tracks
auxilary data
Feature Extraction Unit 1
Feature Extraction Unit 2
Feature Extraction Unit n
features
HMM1
MLM
HMM2
HMMn
Ground Truth
P1
P2
Pn

Figure 3.26: Training and classification using HMMs

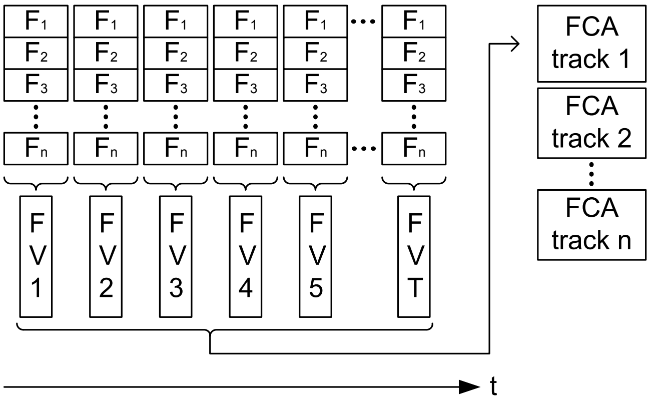


Figure 3.27: (FCA[13]) - The feature structure for training one of the HMMs

Introducing the Decision Tree

The decision tree is a way to display an algorithm that supports decisions [65]. According to [65], decision trees are a way to represent rules which are based on data with hierarchical and sequential structures that recursively partition the data. As its name implies, decision trees use a tree-like graph to show the decisions made and the resulting consequences, including change event outcomes, resource costs and utility [65]. Decision tree learning is an inductive learning algorithm that generates a classification tree for classifying data [65]. Normally, to generate a decision tree a "divide and conquer" strategy is used. The feature space, which is based on a training set is recursively partitioned to generate a classification tree. Thus, a specific decision rule is implemented at each branch, where single or multiple combinations of attribute inputs or features are part of the rule [17].

The structure of a decision tree can be compared to a biological tree. A biological tree consists of roots, branches and leaves - so does the decision tree. Similar to a biological tree that begins with its roots, a decision tree always begins with the root node. The outermost parts of a tree in nature are it's leaves, which is the same for the decision tree. The leaves of the decision tree are called terminal nodes, which represent the final classification [65]. Just as tree sap in a biological tree passes several branches on it's way from the roots to the leaves, the decisions which have to be made in a classification tree also have to pass over several branches. This branches are the interior nodes of a decision tree. They represent the stages leading to the decision [65]. The root and the interior nodes together are also called non-terminal nodes [65]. A set of rules implements the classification process which starts at the root, follows a path through the tree's non-terminal nodes and ends at one terminal node which symbolizes the label of the object being classified [74]. The path through a decision tree is determined by a set of rules. A decision has to be made at each non-terminal node to determine the path to the next node [17]. An example of a simple binary decision tree which meant to help making decisions about selling shares, is given in Figure 3.28 [74].

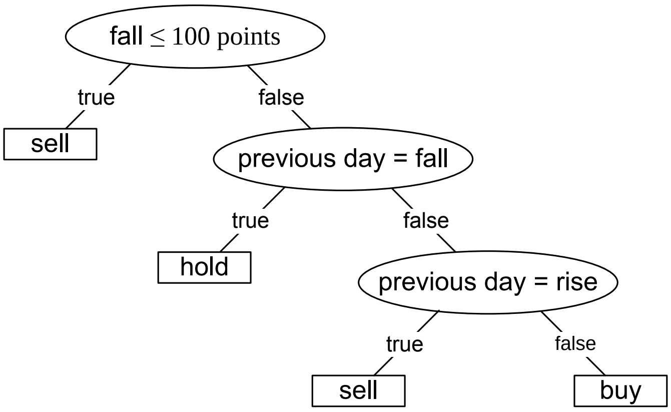


Figure 3.28: A simple decision tree with binary splits (graphics adapted from [74])

According to [65], there is a need for efficient, robust and versatile methods to examine and evaluate data that represents the outcome of experiments and of projects [65]. A decision tree can be used for data exploration in one or in combinations of the following methods [65]:

- **Description:** data can be reduced by transforming it into a more compact form. This compact form is easier to process and better understandable, but the essential characteristics of the data must not get lost. Data in the descriptive form should provide an accurate summary.
- **Classification:** analyzing data and checking whether it contains well-separated classes of objects, in order to interpreted the data in a meaningful way based on a substantive theory.
- **Generalization:** finding a way to map independent to dependent variables to find a model that can forecast future values of the dependent variable.

According to [65], decision trees have a large field of applications. Automatic ways to create rules in the form of decision trees are used in nearly every field that requires data exploration [65]. In the real world decision trees are used in agriculture, astronomy, biomedical engineering, financial analysis, image processing, language processing, manufacturing and production, medicine, etc. [65].

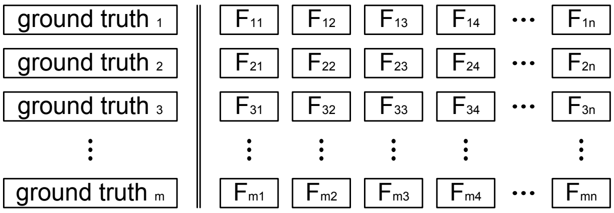
The generation of a decision tree from a training set is called “tree induction” [73], “tree building” or “tree growing” [65]. Typical tree induction systems generate the tree in a greedy top-down fashion. Typically an algorithm similar to the following algorithm is used. The following algorithm starts with an empty tree and uses the a training set to grow the tree: [17] [65]

1. Create a leaf node with the class c if all the training examples at the current node t belong to category s.
2. If not all training examples at the current node t belong to category s, the score of each one of the set of possible splits S by using a goodness measure.
3. The best split $s*$ has to be chosen as test for the current node.
4. For each distinct outcome of $s*$ a child node has to be created. The edges between parent and child nodes have to be labeled with the outcomes of $s*$ and the training data has to be partitioned into the child nodes using $s*$
5. If all training samples at node t belong to the same class as the node, node t is said to be pure. Thus, there is no need to split it in further branches. For all child nodes that are not pure the previous steps have to be repeated.

More details on the way decision trees are generated can be found in [65] and [74]. As the decision tree produces a set of rules for the path trough the tree, the tree can easily be interpreted as a set of if-then instructions [60]. Generally, people are able to understand decision tree models with no or only a little explanation [65]. Decision trees need little data preparation, they are also very flexible with the input data - both numerical and categorical data can be handled [65]. With decision trees it is possible to analyze large amounts of data using standard computing resources [68]. It is a great advantage that there are many approaches to generate a decision tree, as it allows building the tree automatically and then using the simple-to-understand and uncomplicated structure of the tree to implement the set of rules generated by the tree induction to classifying the data [22]. The decision tree therefore promises to be a good way to implement a recognition in an embedded system. [68] [22]

Training and Recognition with Decision Trees

For the gesture recognition introduced in this study, the features introduced earlier in Section 3.6 are used to generate a decision tree, with leaves that represent the decision about gestures resembled by an uncategorized feature set. The features have to be specially prepared for the learning of the decision tree, as the structure used to train the HMMs is not compatible to the data structure for DT training [60]. As depicted in Figure 3.29, the features have to be presented in a table-like structure to the DT induction algorithm. In this structure the ground truth, thus, the name of the actual gesture, has to be stored in the far left column. The training table contains the data for all gestures. No split up has to be made, for only one tree is to be generated for classification. In contrary to the HMM the decision tree is not time dependent. Therefore, the features have to be summarized for each track. This is done by dividing each track into six segments, with the average angle (orientation and orientation from first to current) is calculated for each segment. Separately the average of the angle values of the whole track is used as a feature. The maximum value is used for all other features, resulting in a training data table consisting of seventeen columns - one for the ground truth and sixteen for the features. To recognize or classify a gesture, a feature set calculated from a trajectory is presented to the tree without the ground truth [60]. The decision tree classifier directly presents the name of the gesture stored in each leave. This output can be easily processed or compared to the ground truth for evaluation [60]. In this study the Statistics Toolbox of MATLAB was used to generate a simple binary decision tree. How to use this Toolbox is described in [60].



ground truth $_1$	F_{11}	F_{12}	F_{13}	F_{14}	⋯	F_{1n}
ground truth $_2$	F_{21}	F_{22}	F_{23}	F_{24}	⋯	F_{2n}
ground truth $_3$	F_{31}	F_{32}	F_{33}	F_{34}	⋯	F_{3n}
⋮	⋮	⋮	⋮	⋮		⋮
ground truth $_m$	F_{m1}	F_{m2}	F_{m3}	F_{m4}	⋯	F_{mn}

Figure 3.29: Feature structure for generating the DT

3.8 Cross Validation

For the evaluation of the result of the gesture recognition a leave-one-out cross validation is used. Initially, an equally sized set of recorded tracks for each gesture was randomly picked from the test and training database. It is also possible to select a previously stored data set, so that all evaluations can be made using the same tracks. Another option is to select all data, normally only data sets of equal size are used. The data set is used to calculate the features necessary to train the HMMs or to generate the DT. This is where the leave-one-out process comes into play. Now, all feature sets are used to train the HMMs or to generate the DT except one that is going to be used later to test the classifiers [50]. The single feature set is then presented to the HMM or the DT to be classified and assigned to a certain type of gesture [50]. The result of the classification is then compared to the ground truth. The outcome is then stored in the confusion matrix which is a square matrix that shows the ground truth and actual classifications [50]. The leave-one-out procedure is repeated for all feature sets of the previously selected data set. Therefore, each track is left-out once, while the other tracks are used to train the classifiers. Using the confusion matrix, recognition rates for each single gesture and an overall average recognition rate can be calculated.

3.9 Summary of Methodology

In this chapter the methods used for gesture recognition are introduced and discussed. The first section provides information about the UCOS sensor. As profound knowledge of the underlying technology of the UCOS sensor is required to use this device for gesture recognition such information is provided in Section 3.1. Technical details on the sensor and its current use - counting people - are presented, too. After the introduction of the built-in tracking function of the UCOS sensor the modifications on the AE data format, which are necessary for gesture recognition, are explained.

In Section 3.2 and Section 3.3 two linked topics are introduced. In the first of these sections a concept for a gesture controlled GUI is presented. This GUI is conceived especially for this study. It does not represent a final solution but it gives a clue about which and how many gestures are necessary to be recognized by a gesture recognition system to be used for HCI. Thus, a set of ten gestures - eight directional gestures and two symbolic gestures - were defined in Section 3.3. In the previously presented GUI concept combinations of four, eight or ten of these gestures can be used for navigation.

How these gestures are recorded with the UCOS sensor, which setup and which settings can be used for recording is introduced in Section 3.4. This recorded trajectories are annotated and then stored in a database. This database and the annotation methods are presented in Section 3.5.

The annotated and stored trajectories from the database are used to calculate features for classification. The features discussed in this work are: “lenght”, “speed”, “orientation”, “BB-charactersitics” and “direction changes”. The features were kept simple because only low computational power should be required to calculate them.

The classification methods used in this work are presented in Section 3.7. Because of the promising results in closely related works [42] [67] it was decided that the HMM should be used in this work. The results from this classification method are compared with a second, a rule-based, approach. As a trained DT can be interpreted as a set of rules, this method is selected as second approach. Theory on both methods and how they can be used for gesture recognition are explained, too.

Finally, in Section 3.8 the leave-one-out cross validation is presented as a method for the evaluation of the UCOS-based gesture recognition approach.

CHAPTER 4

Implementation

In this chapter the implementation of the gesture recognition is described. In Section 4.1 general information about the implementation of the training and the recognition process using the HMMs is presented. Information about the implementation of the training and the recognition process using theDTs is presented in Section 4.2. Section 4.3 provides an overview on the implementation of the feature calculation used for HMM and DT.

4.1 Implementation of the HMM classification

As mentioned in Section 3.7, one HMM for each gesture has to be trained. To train a HMM the function `HMM=train(data,M,Q,numb_it,lr,thresh)`[1] from [78] is used. This function trains a HMM for each set of training data. This function requires the training data formatted as described in Section 3.7, the number of states `Q`, the number of Gaussian density functions `M`, the covariance type `cov`, the option for the type of the HMM `left_right` and the threshold for stopping the training `tresh`. In order to find the best result different settings for the number of states and the number of Gaussian density functions were tested. The number of states was varied for every run between 1 and 20. Increasing the number of Gaussian mixtures to values greater than 1 leads to uncontrollable instabilities which have to be ruled out in future work. There are thee different types of covariance possible: “full”, “spherical”’ and “diag” [78]. In this work the setting “diag” was chosen because it led to the best recognition results. Setting `left_right` to 1 enables the so-called left-right HMM. As described in Section 3.7, this type of HMM is chosen for time dependent signals like the gesture use in this work. During the training of the HMM iterations are performed until the model parameters converge with the training data within a certain threshold. This threshold has a default value of 10^{-4}. It was also set to 10^{-3}, 10^{-2} and higher values, where a setting of 10^{-2} proved to deliver the best

---

[1] for better readability shortened: HMMt̄rain(...)

results because convergence problems occurred for small thresholds. Alternatively the training is stopped when the maximum number of iterations - specified by `numb_it` - is reached. Within the function `HMM=train(...)` the functions `init_mhmm(data,Q,M,cov,lr)`[2] and `learn_mhmm(data,prior0,transmat0,mu0,Sigma0,mixmat0,numb_it, thresh,0,cov)`[3] are called, where the `init_mhmm(...)` function initializes the parameters for the function `learn_mhmm(...)` that actually trains the HMMs.

The so trained HMMs recognize gestures by using the recognize function from [78]:
`[result,loglikV,path]=recognize[data,HMM1,HMM2,HMM3...)`[4]
During this process the data is matched to a HMM using the Viterbi Algorithm. The function `recognize(...)` requires the features of the data to be classified and the previously trained HMMs as input. As mentioned above, the output of this function consists of the structures `result` (containing the vector of recognition results), `loglikV` (containing the recognition likelihood) and `path` (containing the optimal path when matching data sequence and model). From this output values, the recognition likelihood is used to classify the track as a certain gesture. The HMM with the maximum likelihood is supposed to match the track.

Paramter	**used setting**
trparam.useequal	true
trparam.tr_quality	2
trparam.no_gestures	10, 8 or 4
featureparam.featureset	[true, true, true, true, false, false, true, true]
featureparam.norm	π
featureparam.multi	1
featureparam.cutfirst	1
featureparam.cutlast	0
featureparam.diffn_orient	3
featureparam.segments	32
featureparam.maxcnt	50
featureparam.tton	0
featureparam.dct_en	true
featureparam.dct_multi	100
hmmparam.Ns	varied between 1 and 20
hmmparam.Kg	1
hmmparam.nr_iter	8
hmmparam.ltr	true
hmmparam.Tresh	1e-2

Table 4.1: Optimal parameters used in the function HMM_chain

The whole process of selecting and loading data from the database, calculating the features, training the HMMs, using them for recognition and performing the cross validation was auto-

---

[2] for better readability shortened: init_mhmm(...)

[3] for better readability shortened: learn_mhmm(...)

[4] for better readability shortened: recognize(...)

mated in the eval function:
`[eval]=HMM_chain(trparam,hmmparam,featureparam,use_track_ids)`[5]. This function selects data from the test database which is then cyclically used for training, recognition and cross valuation. Different settings for the way the tracks are selected, the way the HMM is trained and the way how the features are calculated can be specified, using the input parameters. The parameters for the track settings are specified by the structure `trparam`. The variable `trparam.useequal` can be used to force the function to use training data with an equal amount oft tracks. The variable `trparam.tr_quality` is used to select the quality of the tracks used. The variable `no_gestures` defines the size of the gesture set used, where "10" represents the full gesture set and "4" represents the minimum useful gesture set. The variable `hmmparam` contains the earlier described parameters used to train the HMMs. The feature parameters are used to specify which features should be calculated and how they should be calculated. Using the variable `feAtureparam.featureset`, the features can be turned on and off by setting the referring values to "true" or "false". The setting `[true, true, true, true, false, false, true, true]` was used in this work to activate all features except "speed" and "length". The parameter `featureparam.norm` can be used to activate the normalization of the feature values. The feature values for the orientations are in a well knowN range between $-\pi$ and $+\pi$. "BB-area" and "BB-ratio" are determined by the resolution of the sensor. As the UCOS sensor has a resolution of 128 x 128 pixels the maximum value for "BB-ratio" is 128 and the maximum value for the "BB-area" is 16384. The count of the directions is limited by the value `featureparam.maxcnt`. All these values of the features are normalized between 0 and the value defined by the parameter `featureparam.norm`. If `featureparam.norm` is set to "0", the normalization is turned off. The values of the features can also be multiplied by a certain value which is specified by the parameter `featureparam.multi`. With the parameters `featureparam.cutfirst` and `featureparam.cutlast` one can specify whether and how many elements should be cut away at the beginning and the end of each feature vector. Typically, the first element of the feature vector is cut away because it is in most cases 0, while no cuts are made at the end. The parameter `featureparam.tton` is used to enable and disable the original time stamp in the feature vector. The time stamp is not required for training the HMM, but it is useful for the evaluation of the features. The parameter `featureparam.dct_en` is used to enable or disable the performing of a discrete cosine transformation on the feature vector and the parameter `featureparam.dct_multi` is used to multiply the result of the DCT[6] by a value specified in this parameter [42]. The parameter `use_track_ids` is used to hand a set of track IDs to the function. The track IDs are then used for training and classification, instead of randomly picked IDs. The output `[eval]` of the function `HMM_chain(...)` contains the confusion matrix from the cross validation (`eval.Confusion`), the IDs of the tracks used for training and recognition (`eval.track_ids`), the IDs of the wrong classified tracks (`eval.false_ids`) and the input parameters (`eval.hmmparam`,`eval.featureparam` and `eval.trparam`). The input parameters and the track IDs stored in the output can be used as an input for further test runs. An overview of the parameter settings for the function `HMM_chain(...)` is given

---

[5] for better readability shortened: HMM_chain(...)

[6] Discrete Cosine Transformation

in Table 4.1. The code of the function `HMM_chain(...)` is listed in Appendix B.

4.2 Implementation of the DT (Quasi Rule Based) Classification

In contrast to the HMM, only one decision tree has to be generated for the gesture set. The input data for training has to contain all information on gestures that have to be recognized. The decision tree is generated with the function `tree=classregtree(Train.F, Train.gt, 'names',Data.fname)` which is included in the Statistics Toolbox from MathWorks [60]. The input for this function consists of the features from the training data (`Train.F`) and the corresponding ground truth (`Train.gt`) in the format presented in section 3.7. The parameter `'names'` indicates that the value of `Data.fname` is a cell array of names for the predictor variables (features) in the same order in which they appear in the feature table from which the tree is created.

The tree created by the previously described function is used for classification with the function `result=eval(tree,Test.F)`[7] which is part of the Statistics Toolbox from MathWorks, too [60]. This function requires the previously generated tree (`tree`) and the features of the data to be classified (`Test.F`) as its input. The function `eval(...)` returns the names of the gestures as classification result in the same order as presented to the decision tree.

Paramter	**used setting**
trparam.useequal	true
trparam.tr_quality	2
trparam.no_gestures	10, 8 or 4
featureparam.featureset	[true, true, true, true, false, false, true, true]
featureparam.norm	0 (means off)
featureparam.multi	1
featureparam.cutfirst	1
featureparam.cutlast	0
featureparam.diffn_orient	3
featureparam.segments	32
featureparam.maxcnt	50
featureparam.tton	0
featureparam.dct_en	false
featureparam.dct_multi	1

Table 4.2: Optimal parameters used in the function DT_chain

Similar to the HMM, the whole process of selecting and loading data from the database, calculating the features, generating the DT, using it for classification and performing the cross validation was automated with the function:
`[retvals]=DT_chain(trparam,featureparam,use_track_ids)`[8].

---

[7] for better readability shortened: eval(...)

[8] for better readability shortened: DT_chain(...)

Basically, this function does the same as the function `HMM_chain(...)` that was described in Section 4.1. It selects data from the test database or uses predefined data for cyclical training, recognition and cross validation. The input parameters are more or less the same as for the function `HMM_chain(...)`. However, the DT does not need `hmmparam`- only `trparam` and `featureparam` have to be specified. The parameters for the track selection are the same as for the HMM. The same tracks are used for both classification methods to achieve a comparable result. The main difference between these two approaches is due to the parameters for the feature calculation. In case of the DT the normalization (`featureparam.norm`) is turned off and the multiplier for the features (`featureparam.multi`) is always set to 1. Also the DCT is turned off by setting the parameter (`featureparam.dct_en`) to "false". Therefore, the parameter (`featureparam.dct_multi`) has no meaning and is set to 1. The output `[retvals]` of the function `DT_chain(...)` contains the confusion matrix from the cross validation (`retvals.Confusion`), all decision trees generated during the cross validation `retvals.treecollection` and the input parameters (`eval.featureparam` and `eval.trparam`). An overview of the parameter settings for the function `DT_chain(...)` is given in the Table 4.2. The code for the function `DT_chain(...)` can be found in Appendix B.

4.3 Feature Calculation

For the calculation of each of the features described in Section 3.6 a function was implemented. These functions primary use the calculations that are described in Section 3.6. The parameters that are stored in `featureparam` are used to calculate the features from the AE-data. The variable `featureparam` is explained in Section 4.2 and Section 4.1. The parameter `featureparam. norm` is optional for all functions. As described above, this parameter can be used to specify a normalization of the calculated feature data. The AE data which is used to calculate the features is handed over to the feature calculation function by the input parameter `Tr`. For the calculation of the feature "orientation between first and current point" the function `Orientation1tE=f_trorient_1toend(Tr,featureparam.norm)` is used.
The function `Orientation=f_trorient(Tr,diffn_or,featureparam.norm)` is used to calculate the feature "orientation". The parameter `diffn_or` specifies the distance between two corresponding points of a trajectory which are used to calculate the orientation. The function `[BB-area,BB-ratio]=f_trBB(Tr,featureparam.norm)` is used to calculate the BB features. This function returns the features "BB-area" and "BB-ratio". The function `[chX,chY]=f_trdirchange(Tr,segments,maxcnt,featureparam.norm)` is used for the calculation of the direction changes. The parameter `maxcnt` specifies the maximum number of direction changes to be counted.

For increased flexibility the function `FV=featcollector3(Tr,timemstmp_on, featureparam,diffn_orient,segments,maxcnt)` is implemented. This offers the option to generate a feature set according to the parameter `fetureparam.featureset` that is described in Section 4.1 above.

4.4 Summary of Implementation

In this chapter the implementation of the training and the recognition process using the HMMs, the implementation of the training and the recognition process using the DT and the implementation of the feature calculation were presented. All of the functions introduced are very flexible and offer options for parameter variations. Thus, they might be useful for future use, too.

CHAPTER 5

Experimental Results

In this chapter the results of the evaluation of the gesture recognition methods evaluated in this thesis are presented. In Section 5.1 a visual inspection of the features used in this work is presented.

5.1 Visual Evaluation of Features

Before training the HMM or the decision tree, the calculated features have to be evaluated. This is done by performing a visual inspection of the calculated features. The features “orientation”, “orientation from the beginning to the current point”, “speed”, “length”, “BB-area”, “BB-ratio”, “horizontal directional changes” and “vertical directional changes” are calculated for each recorded track that symbolizes a gesture. All results of the calculations for a certain gesture type are plotted into figures - one figure for each feature. For each kind of gesture the results of one type of feature are plotted one after another resulting in a side by side view that enables a visual inspection whether the feature has distinctive characteristics. The results are presented in eight plots, one for each feature. With these plots the characteristics of the ten gestures defined in Section 3.3 can be evaluated. This overview makes it easy to see whether the calculated features show significant differences for the different gestures. It is assumed that the more the features are different for each gesture the better the recognition results are. For example, the comparison of the feature “orientation between first and current point” is displayed in Figure 5.1 in this section. The comparisons for the other features can be found in Appendix beginning on page 97.

As visible in Figure 5.1 the “orientation between first and current point” is significantly different for all gestures. The accumulation of the angle values for “down” is about π and $-\pi$. The accumulation of the angle values for “up” is about 0. It could be observed, that the accumulations for “left” and “right” are about $\frac{\pi}{2}$ and $\frac{-\pi}{2}$. For the diagonal gestures “rightup”, “leftup”, “leftdown” and “rightdown” the accumulations of angles are about $\frac{-\pi}{4}$, $\frac{\pi}{4}$, $\frac{3\pi}{4}$ and $\frac{-3\pi}{4}$. For the gesture roof no clear accumulation is visible, but the values seem to change in a characteristic way between $\frac{-\pi}{2}$ and 0. The feature "orientation between first and current point" is inconclusive for the "wave" gesture, but it shows a significant accumulation near $\frac{-\pi}{2}$. The fact that the gestures used in this work were performed by a right handed person could provide an explanation for this accumulation. As visible in Figure A.1 the same accumulations as for "orientation between first and current point" can be found in the accumulations of the of the feature "orientation", but "orientation between first and current point" has a more averaging effect. Being very characteristic for nearly all gestures, the two "orientation" features are used for gesture recognition.

Comparisons of the calculated features "length" and "speed" for each gesture type track are depicted in Figure A.3 and Figure A.2. As visible in these illustrations, neither "length" nor "speed" show remarkable differences for the different gestures. Therefore, these two features are not relevant for the gesture recognition.

When looking at the BB features, respectively the "BB-area" depicted in Figure A.4 and the "BB-ratio" depicted in Figure A.5, different behaviors for different gestures are visible. The "BB-area" value remains relatively low for the horizontal and vertical gestures. It increases fast to high values for the diagonal gestures. For the gesture "roof", the development of "BB-area" is intermediate between the development of the vertical/horizontal "BB-area" and the development of the diagonal "BB-area". Therefore, it is slightly different to both. Overall, the "BB-area" for the gesture "wave" stays on the lowest values, but the difference is not so significant in comparison to the horizontal/vertical gestures. The "BB-ratio" shows more differences between the gestures. It increases fast for the "left" and "right" gestures, but it stays nearly at 0 in the case of "up" and "down" gestures. It is also visible that the "BB-ratio" in the case of diagonal gestures accumulates around 1. The "BB-ratio" for the gesture "roof" is like a mix of diagonal and horizontal gestures. It develops very characteristically over time. The "BB-ratio" for the gesture "wave" has a significant accumulation around 0.5. These facts lead to the conclusion that the BB-characteristics are very important features for the gesture recognition application examined in this work.

The calculation results of the features that count the direction changes in horizontal and vertical directions are depicted in Figure A.7 and Figure A.6. As "waving" consists of movements into different directions these two features - "vertical direction changes" and "horizontal direction changes" - are significantly different for the gesture "wave" in comparison to the other gestures. For the gesture "roof" the number of direction changes is slightly higher than for all other gestures except wave. As the directional features are characteristic for the gestures "roof" and "wave" these features are to be used for gesture recognition, too.

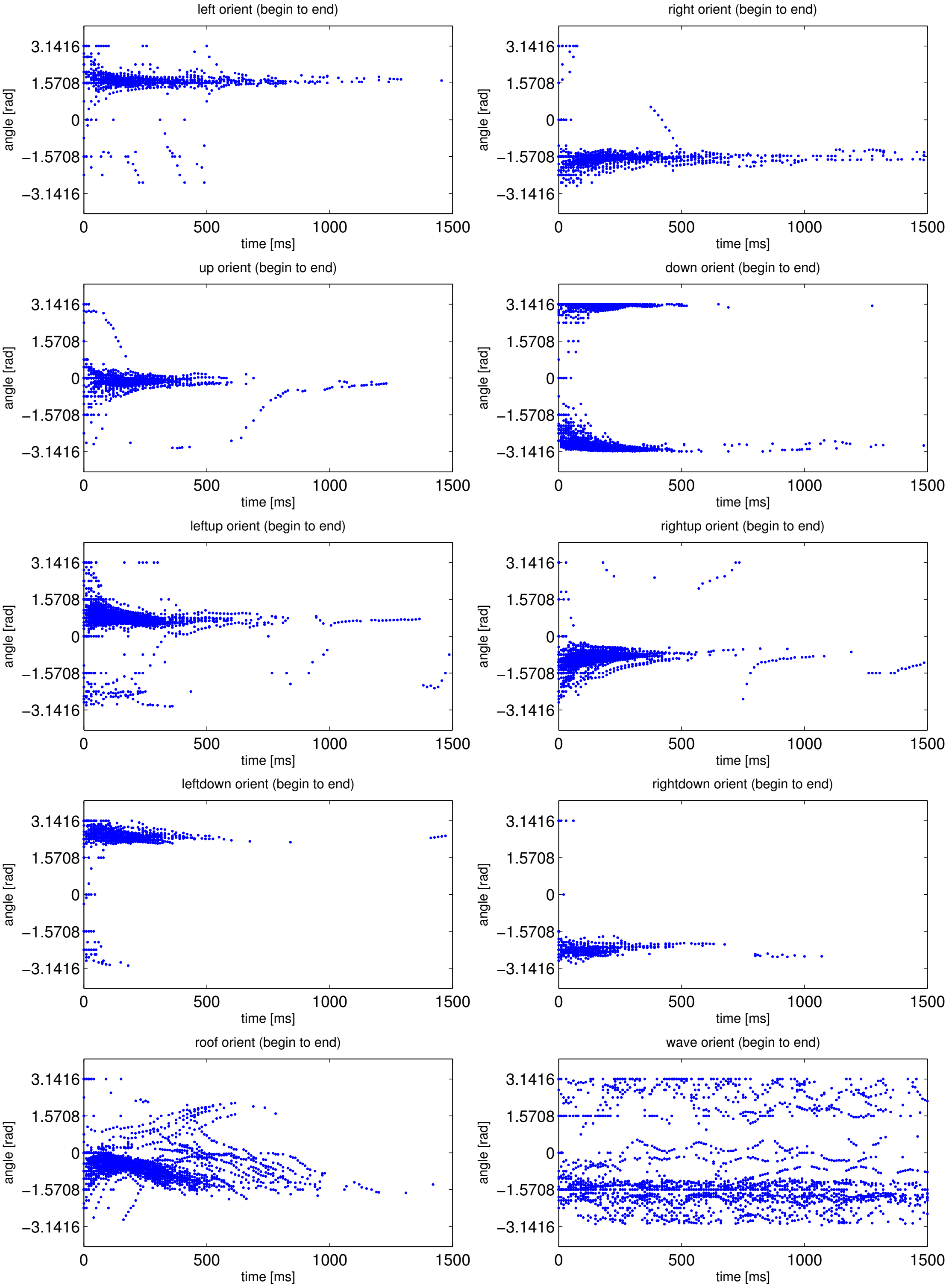


Figure 5.1: Comparison of the feature "orientation between first and current point" for ten different gestures

5.2 Preface for Classification

The HMMs and the DT were trained with annotated data. This data was recorded from one person performing the gestures defined in Section 3.3 and by using the test setup presented in Section 3.4. The recorded data was categorized as gestures with different quality measures according to the visual inspection of the recorded data. The quality criteria are divided into five grades: “bad (1)”, “acceptable (2)”, “good (3)”, “very good (4)” and “pefect (5)”. Unrecognizable and unintentional movements were sorted out of the recorded data and tagged “invalid (0)”. All results were stored in a database containing 1463 recorded tracks of which 141 are “invalid (0)”, therefore, not used for training and recognition. The tracks with the quality measure “bad (1)” were not used, too. The unused data is kept in the database for future evaluation. Tracks rated as “acceptable (2)” and better were randomly selected for the initial training of the HMM and the DT. For better comparability of the results, the same tracks were used for other training and recognition runs with other training parameters, given that the same size of the gesture set was used. To provide a better overview the data sets for each gesture were kept at the same size. When a gesture set of ten or eight gestures is used, the gesture “leftup” with the smallest number of 52 valid samples determines the size of the training data sets for the other gestures to 52 tracks. As the gesture “leftup” is not part of the minimal gesture set of four gestures, the size of the data set for each gesture is determined by the 89 valid database entries for the gesture “left”. Therefore, the tests with the reduced gesture set were done with 89 tracks for each gesture. Table 5.1 gives an overview on the status of the training database.

DB entries	**bad**	**acceptable**	**good**	**very good**	**perfect**	**invalid**	**sum**
down	18	48	102	20	4	0	192
up	52	35	43	12	2	0	144
left	24	36	41	9	3	0	113
right	10	25	52	19	7	0	113
rightup	50	59	60	13	1	0	183
leftup	11	24	17	11	0	0	63
rightdown	28	57	99	20	0	0	204
leftdown	13	24	43	9	1	0	90
roof	28	52	63	10	2	0	155
wave	12	17	24	12	0	0	65
invalid	0	0	0	0	0	141	141
total							**1463**

Table 5.1: Status of the test and training database

5.3 Classification with the Hidden Markov Models

This section summarizes the classification results of the HMMs. The best recognition result for the full gesture set of ten gestures, the best result for a set consisting of eight gestures and the best recognition result for a minimum gesture set with only four gestures are introduced on the following pages.

Classification using the HMM on ten gestures

The full gesture set used for the classification in this work consists of the gestures “left”, “right”, “up”, “down”, “rightup”, “rightdown”, “leftup”, “leftdown”, “roof” and “wave”. The parameters for the evaluation were set as specified in Table 4.1. The number of states for the HMM was varied between 1 and 20, the result of this variation can be seen in Figure 5.2. In this figure it can be seen that the best result with ten gestures is achieved when setting the number of states to 14. At this point, both, the blue line of the average recognition and the lower red line for the minimum recogniton rate (the rate for the worst recognized gesture) are at their maximum. For other settings the upper red line, which represents the maximum recognition rate for a single gesture, reaches 100%, but the minimum and average recognition rates at this settings are much worse. Therefore, the best result for recognizing ten gestures was achieved using 14 states for the training of the HMM.

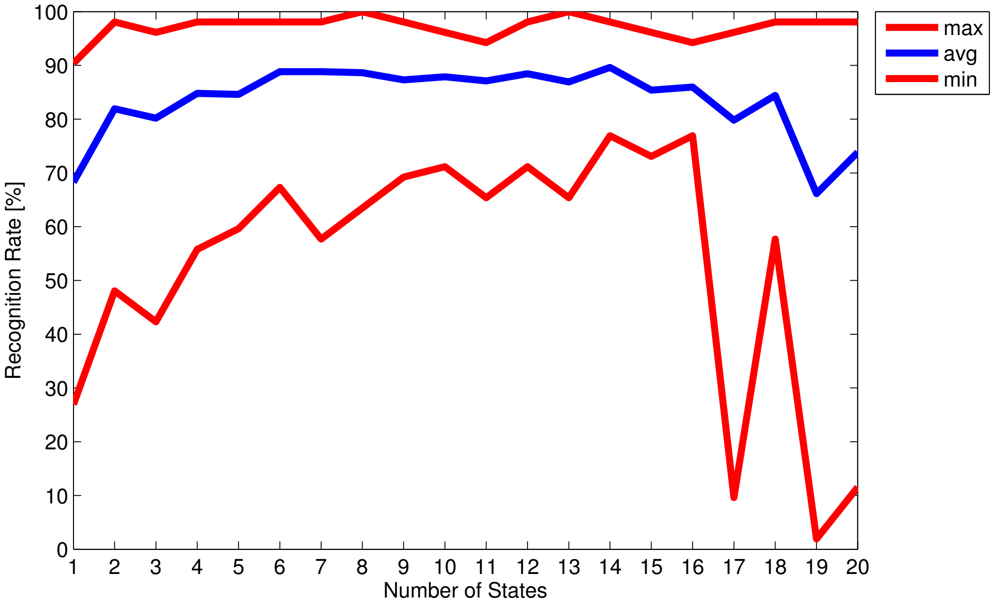


Figure 5.2: HMM: finding the best result for the parameter “number of states” for ten gestures

Figure 5.3 and Table 5.2 show the results of the cross validation for the best result with ten gestures. The average recognition rate using ten gestures is 89.61%. It is clearly visible that the gestures "rightup" and "rightdown" have a poor recognition rate of 76.92% because they are often mismatched with the gesture "roof" and "down". This happens because the gesture "roof" consists of a movement similar to "rightup" combined with "rightdown". The movement for the gesture "rightdown" is very similar to the movement for "down", because the gestures are not always performed in a perfect way. The gesture "down" has a relatively high recognition rate of 96.15%, however 20.63% of the gestures classified as "down" are classified false. The gesture "roof" has with its 18.97% a relatively high rate of false classifications. This result correspond with the low classification rates of the gestures "rightup" and "rightdown". The gesture "wave" is also a relatively problematic gesture being often classified as another gesture. The recognition rate for "wave" is 80.77%, all other gestures except "rightup" and "rightdown" have a recognition rates greater than 90%.

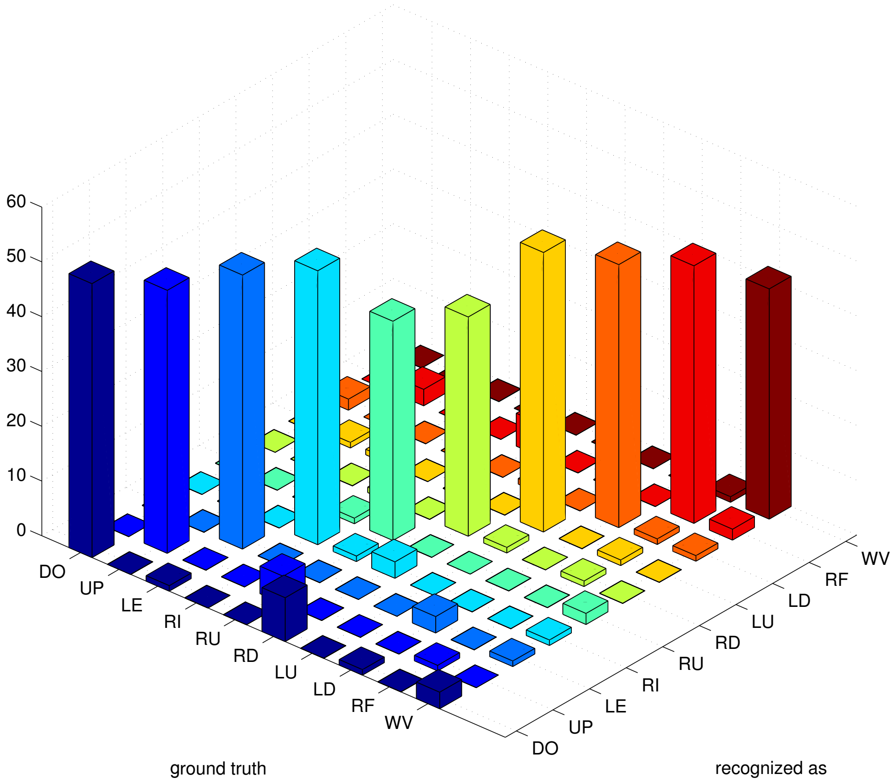


Figure 5.3: HMM: recognition summary of the best result for ten gestures

	DO	**UP**	**LE**	**RI**	**RU**	**RD**	**LU**	**LD**	**RF**	**WV**	**T**	**F**	**TPR [%]**
DO	50	0	0	0	0	0	0	2	0	0	50	2	96.15
UP	0	48	0	0	0	0	1	0	3	0	48	4	92.31
LE	1	0	50	0	0	0	1	0	0	0	50	2	96.15
RI	0	0	0	50	1	1	0	0	0	0	50	2	96.15
RU	0	5	0	1	40	0	0	0	6	0	40	12	76.92
RD	8	0	0	3	0	40	0	1	0	0	40	12	76.92
LU	0	0	0	0	0	1	51	0	0	0	51	1	98.08
LD	1	0	3	0	0	0	0	48	0	0	48	4	92.31
RF	0	1	0	0	0	1	1	1	47	1	47	5	90.38
WV	3	0	1	1	2	0	0	1	2	42	42	10	80.77
T	50	48	50	50	40	40	51	48	47	42			
F	13	6	4	5	3	3	3	5	11	1		**AVG**	89.61
FPR [%]	20.63	11.11	7.41	9.09	6.98	6.98	5.56	9.43	18.97	2.33		9.85	

Table 5.2: HMM: confusion matrix of the best result for ten gestures

Classification using the HMM on eight gestures

To rule out the problems considering the recognition of the gestures “wave” and “roof”, these gestures were left out and the classification was performed with only eight directional gestures. Again the parameters for the evaluation were set as specified in Table 4.1 and the number of states were varied between 1 and 20. One can read from Figure 5.4 that the best result for the state variation with eight gestures is achieved when the Number of States is set to eight. The blue line of the average recognition rate is at the maximum compared to the other points as well as the lower red line for the minimum recognition rate. As for ten gestures, other settings could lead to a maximum recognition rate for a single gesture of 100%, but the minimum and the average recognition rates at other settings are much lower. The best result for the recognition of the eight gestures is achieved when using eight states.

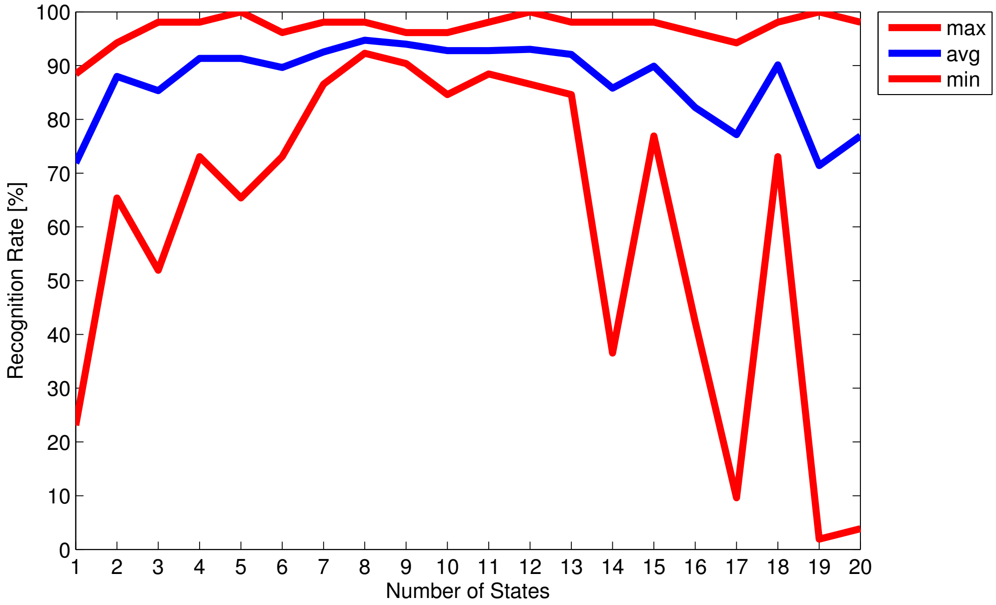


Figure 5.4: HMM: finding the best result for the parameter “number of states” for eight gestures

Figure 5.5 and Table 5.3 show the result of the cross validation for the best result with 8 gestures. The average recognition rate in the case of using eight gestures is 94.71%. Having left out the trouble makers “roof” and “wave”, the highest false classification rate of 7.55% can still be found at the gesture “down”, which is primary mismatched with the gesture “rightdown”. Also, the gesture “rightdown” on its part has a false classification rate of 7.55% being mismatched with the gesture “leftup”. This could be caused by the small unintentional upwards movement of the arm after performing the intentional downwards movement. The gesture “rightup” and “rightdown” have the poorest recognition rate of 92.31%, while all other gestures have a recognition rate greater than 94%. This result leads to the conclusion that the features selected for the gesture recognition are more distinguishable for the classifier when using a smaller set of more simple gestures, leading to a better result in comparison to the recognition of ten gestures.

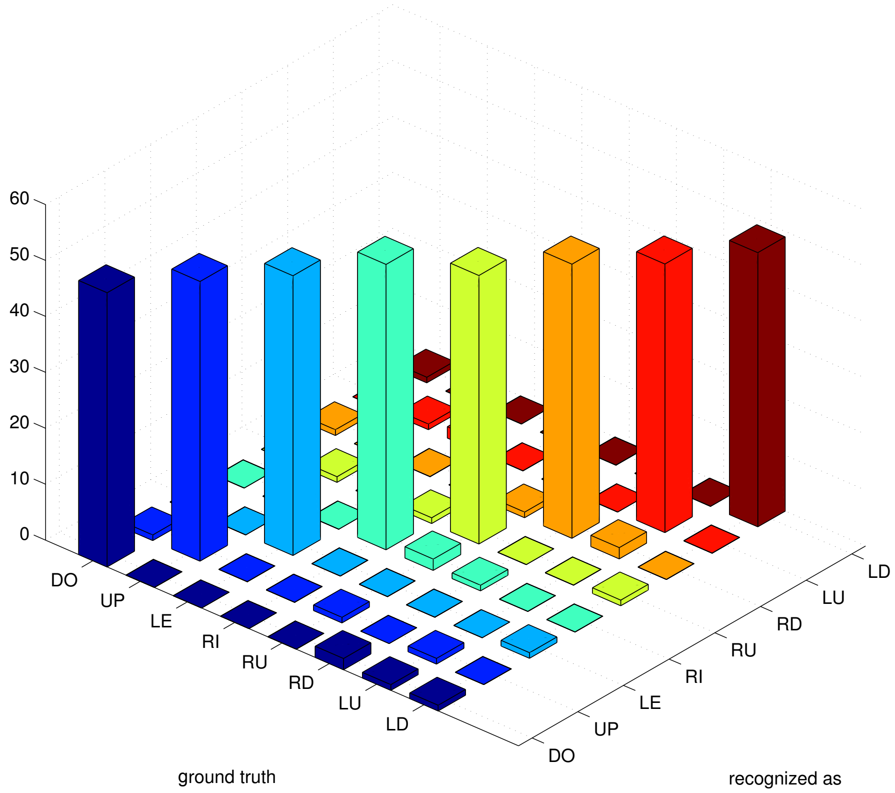


Figure 5.5: HMM: recognition summary of the best result for eight gestures

	DO	**UP**	**LE**	**RI**	**RU**	**RD**	**LU**	**LD**	**T**	**F**	**TPR [%]**
DO	49	1	0	0	0	1	0	1	49	3	94.23
UP	0	50	0	0	1	0	1	0	50	2	96.15
LE	0	0	50	0	0	0	2	0	50	2	96.15
RI	0	0	0	51	1	0	0	0	51	1	98.08
RU	0	1	0	2	48	1	0	0	48	4	92.31
RD	2	0	0	1	0	49	0	0	49	3	94.23
LU	1	1	0	0	0	2	48	0	48	4	92.31
LD	1	0	1	0	1	0	0	49	49	3	94.23
T	49	50	50	51	48	49	48	49			
F	4	3	1	3	3	4	3	1		**AVG**	94.71
FPR [%]	7.55	5.66	1.96	5.56	5.88	7.55	5.88	2.00		5.26	

Table 5.3: HMM: confusion matrix of the best result for eight gestures

Classification using the HMM on four gestures

As there occur still relatively high rates of false classifications when using eight gestures, the gesture set was reduced to a minimum gesture set. Only the four directional gestures “left”, “right”, “up” and “down” were used for the third evaluation of the HMM-based gesture recognition. As the visual inspection of the features has showed this four are very distinguishable. The parameters for the evaluation were again set as specified in Table 4.1 and the number of states was varied between 1 and 20. The best setting for the number of states can be found in Figure 5.6. The best result for the state variation with 4 gestures is achieved when the number of states is set to 4,5 or 16. At this settings the recognition rate for all gestures is 100% - the lines for the minimum, the maximum and the average recognition rate meet at 100%. For example, the result with a setting of 5 states is presented here. However, the result is exactly the same with 4 or 16 states.

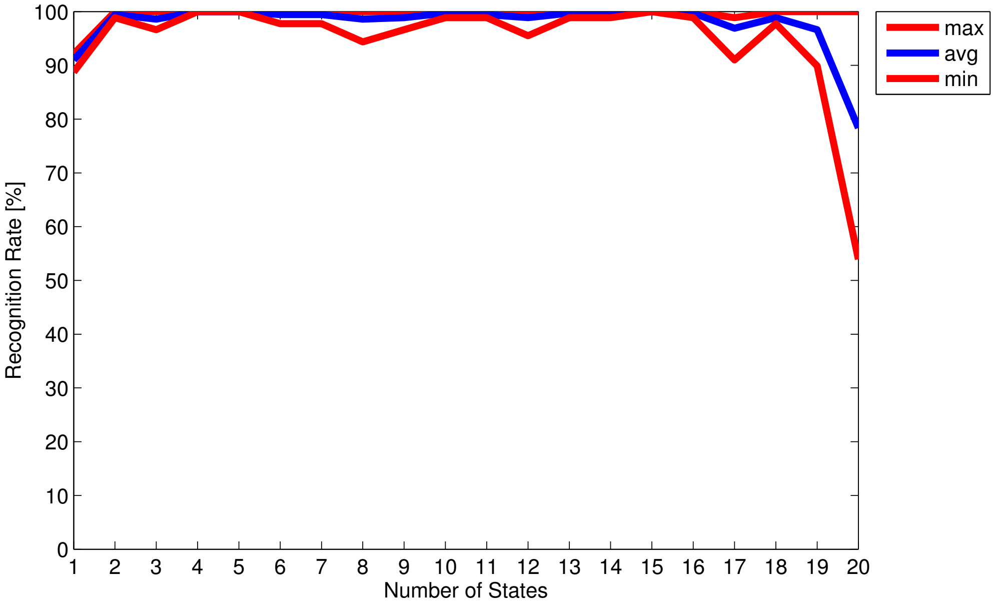


Figure 5.6: HMM: finding the best result for the parameter “number of states” for four gestures

Figure 5.7 and Table 5.4 show the result of the cross validation for the best result with four gestures. The average recognition rate when using four gestures is 100.00%. Reducing the gesture set leads to very good results. Having left out all of the problematic gestures, no gestures are mismatched, as the features - especially the angles - are very different if only four directions are relevant. The reduced gesture set tolerates poorly executed movements that could easily be interpreted as another gestures, if there are more than four gestures to recognize. To summarize, a gesture set of four gestures leads to more accurate result compared to the results of the recognition of eight or ten gestures.

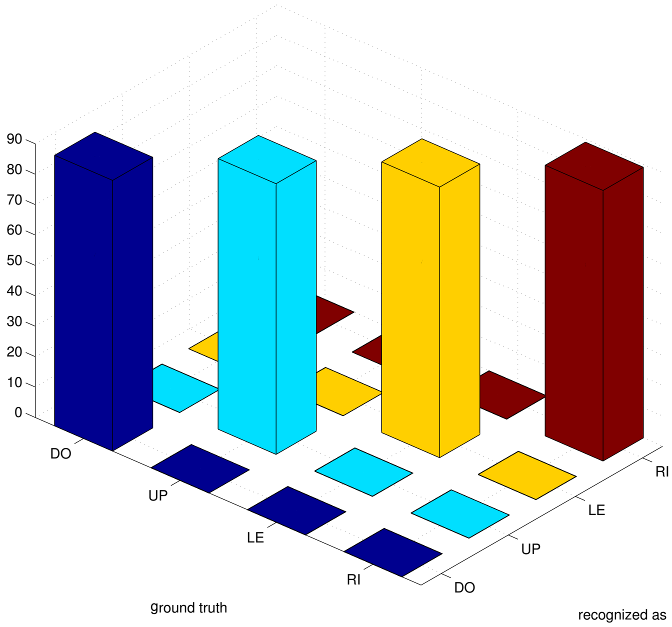


Figure 5.7: HMM: recognition summary of the best result for four gestures

	DO	**UP**	**LE**	**RI**	**T**	**F**	**TPR [%]**
DO	89	0	0	0	89	0	100.00
UP	0	89	0	0	89	0	100.00
LE	0	0	89	0	89	0	100.00
RI	0	0	0	89	89	0	100.00
T	89	89	89	89			
F	0	0	0	0		**AVG**	100.00
FPR [%]	0.00	0.00	0.00	0.00		0.00	

Table 5.4: HMM: confusion matrix of the best result for four gestures

5.4 Classification with the Decision Tree (Quasi Rule Based)

This section summarizes the classification results of the DT. The best recognition result for the full gesture set of ten gestures, the best result for a set consisting of eight gestures and the best recognition result for a minimum gesture set with only four gestures are introduced in the following sections.

Classification using the DT on ten gestures

Similar to the classification using the HMM, the full gesture set used for this classification approach consists of the gestures “left”, “right”, “up”, “down”, “rightup”, “rightdown”, “leftup”, “leftdown”, “roof” and “wave”. The parameters for the evaluation are set as specified in Table 4.2. Figure 5.8 and Table 5.5 show the best result of the cross validation for the recognition of ten gestures. When using these ten gestures the average recognition rate is 91.15%. The gestures “roof” and “wave” are the most problematic, because both of them have a high false classification rate - greater than 19%. Especially, the recognition of the gesture “wave” is with 19.61% often wrong as it is frequently false categorized as the gestures “right” and “roof”. This increases the false classification rate for “right” to 9.26% and drops recognition rate for “wave” to 78.85%. In addition to “wave” also the gestures “roof” and “leftdown” are often not classified the way the should be. While these three gestures have a recognition rate below 90%, the other seven gestures have a recognition rate greater than 90%.

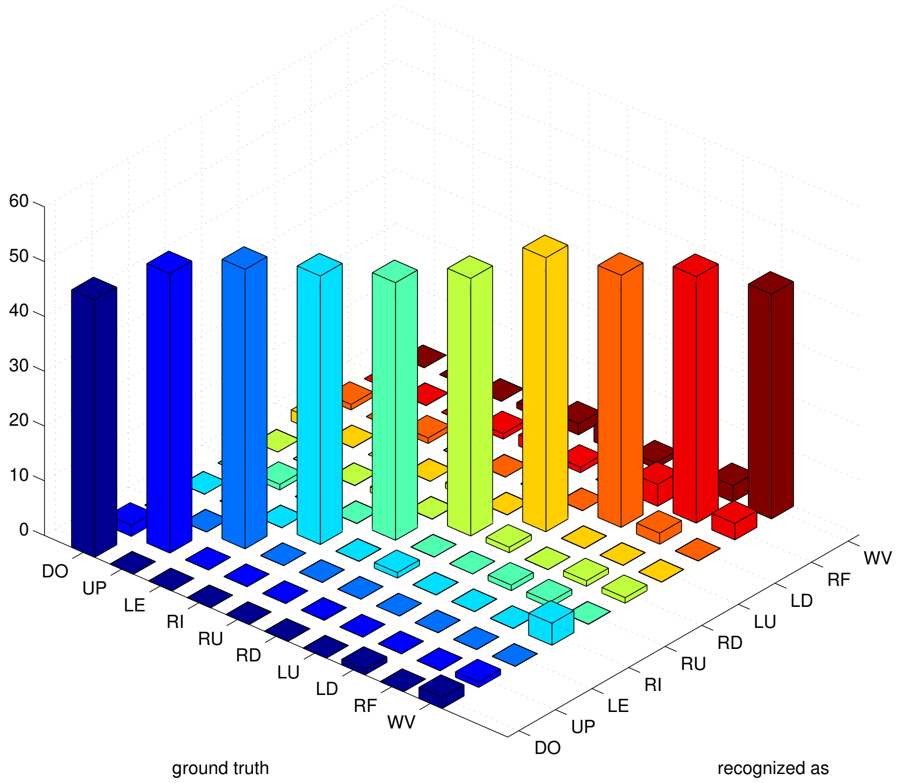


Figure 5.8: DT: recognition summary of the best result for ten gestures

	DO	**UP**	**LE**	**RI**	**RU**	**RD**	**LU**	**LD**	**RF**	**WV**	**T**	**F**	**TPR [%]**
DO	47	2	0	0	0	0	2	1	0	0	47	5	90.38
UP	0	51	0	0	1	0	0	0	0	0	51	1	98.08
LE	0	0	51	0	0	0	0	1	0	0	51	1	98.08
RI	0	0	0	49	0	1	0	0	1	1	49	3	94.23
RU	0	0	0	0	47	0	1	0	2	2	47	5	90.38
RD	0	0	0	1	0	47	0	0	1	3	47	5	90.38
LU	0	0	0	0	0	1	50	0	0	1	50	2	96.15
LD	1	0	0	0	1	0	0	46	4	0	46	6	88.46
RF	0	0	0	0	1	1	0	2	45	3	45	7	86.54
WV	2	1	0	4	0	1	0	0	3	41	41	11	78.85
T	47	51	51	49	47	47	50	46	45	41		**AVG**	91.15
F	3	3	0	5	3	4	3	4	11	10			
FPR [%]	6.00	5.56	0.00	9.26	6.00	7.84	5.66	8.00	19.64	19.61		8.76	

Table 5.5: DT: confusion matrix of the best result for ten gestures

Classification using the DT on eight gestures

As one can see in the result for ten gestures “wave” and “roof” are the most problematic gestures to be recognized as such. These gestures were therefore left out and the classification was performed with only eight directional gestures with the parameter settings as specified in Table 4.2. Figure 5.9 and Table 5.6 show the best result of the cross validation with a gesture set that consist of eight gestures. The average recognition rate using the reduced gesture set is 96.39%. Similar to the HMM leaving out the problematic gestures “roof” and “wave” lowers the maximum false classification rate to 7.41%. No gesture was falsely recognized as the gestures “leftdown” and “left”. The gesture “left” has even a recognition rate of 100%. Similar to the HMM, the features selected for the gesture recognition are more distinguishable for the classifier when using a smaller set of gestures, which leads to more accurate results than in the case of a bigger gesture set of ten gestures.

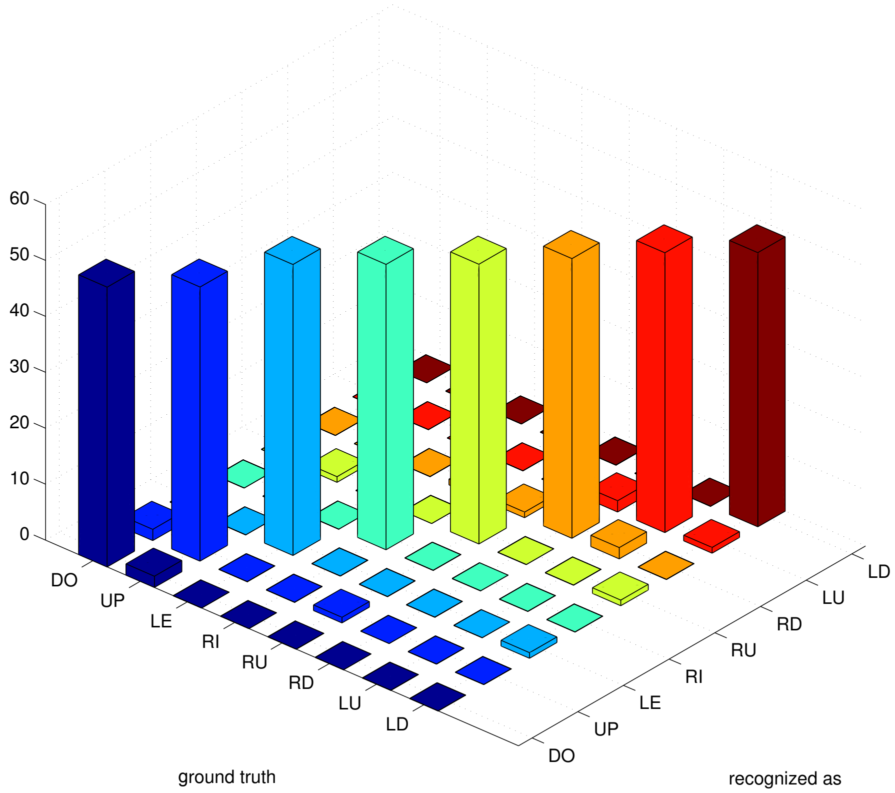


Figure 5.9: DT: recognition summary of the best result for eight gestures

	DO	**UP**	**LE**	**RI**	**RU**	**RD**	**LU**	**LD**	**T**	**F**	**TPR [%]**
DO	50	2	0	0	0	0	0	0	50	2	96.15
UP	2	49	0	0	1	0	0	0	49	3	94.23
LE	0	0	52	0	0	0	0	0	52	0	100.00
RI	0	0	0	51	0	1	0	0	51	1	98.08
RU	0	1	0	0	50	1	0	0	50	2	96.15
RD	0	0	0	0	0	50	2	0	50	2	96.15
LU	0	0	0	0	0	2	50	0	50	2	96.15
LD	0	0	1	0	1	0	1	49	49	3	94.23
T	50	49	52	51	50	50	50	49			
F	2	3	1	0	2	4	3	0		**AVG**	96.39
FPR [%]	3.85	5.77	1.89	0.00	3.85	7.41	5.66	0.00		3.68	

Table 5.6: DT: confusion matrix of the best result for eight gestures

Classification using the DT on four gestures

Similar to the classification using the HMMs, the gesture set was reduced to a minimal gesture set of four directional gestures. The parameters for the evaluation were kept as specified in Table 4.2. The best result of the cross validation with four gestures is shown in Figure 5.10 and Table 5.7. The average recognition rate when using only four gestures is 99.16%. The gesture "up" was only once classified as "down" and the gesture "down" is falsely classified twice as "up". This reduced the recognition rate for "down" to 98.88% and the recognition rate for "up" to 97.75%. The other two gestures have a recognition rate of 100.00%. Removing all problematic gestures from the gesture set leads to very accurate results.

Again, the features - especially the orientations - are very different for four directions. The reduced gesture set once more seems to tolerate poorly executed movements better. As for the HMM, the recognition rate in case of using four gestures is significantly higher when using the reduced gesture set of four gestures than when eight or ten gestures have to be classified.

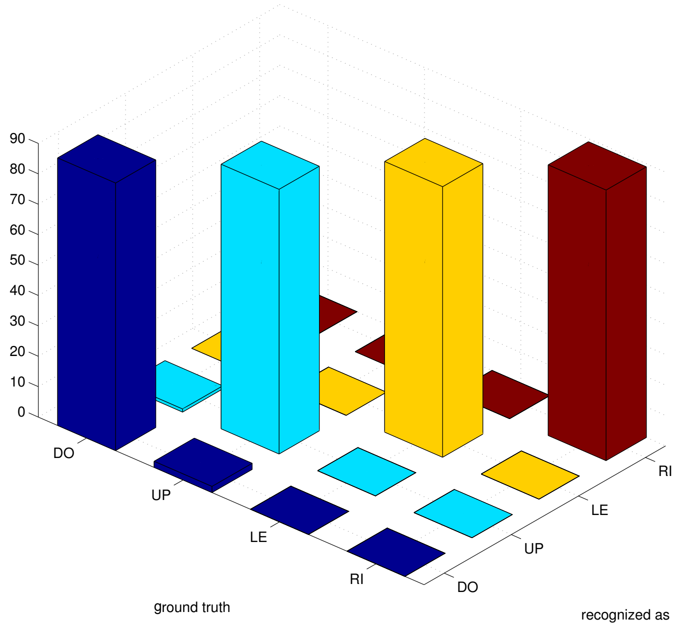


Figure 5.10: DT: recognition summary of the best result for 4 gestures

When using four gestures, the rules generated for classifying the data are easy to read and to transfer to other programming languages. The MATLAB generated code of the decision tree (see Listing 5.1) for four gestures consists of just 13 lines - primary simple if-then instructions. A sample of the decision tree for four gestures can be seen in Figure 5.11.

	DO	**UP**	**LE**	**RI**	**T**	**F**	**TPR [%]**
DO	88	1	0	0	88	1	98.88
UP	2	87	0	0	87	2	97.75
LE	0	0	89	0	89	0	100.00
RI	0	0	0	89	89	0	100.00
T	88	87	89	89			
F	2	1	0	0		**AVG**	99.16
FPR [%]	2.22	1.14	0.00	0.00		0.84	

Table 5.7: DT: confusion matrix of the best result for 4 gestures

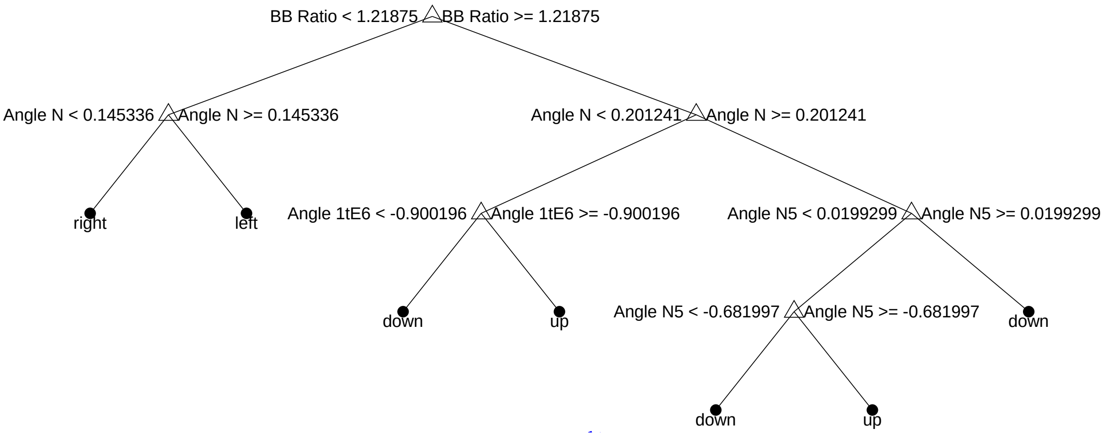


Figure 5.11: Example of trained decision tree for 4 gestures

```
Decision tree for classification
 1  if BB Ratio<1.21875 then node 2 elseif BB Ratio≥1.21875 then node 3 ...
     else up
 2  if Angle N<0.145336 then node 4 elseif Angle N≥0.145336 then node 5 ...
     else left
 3  if Angle N<0.201241 then node 6 elseif Angle N≥0.201241 then node 7 ...
     else up
 4  class = right
 5  class = left
 6  if Angle 1tE6<-0.900196 then node 8 elseif Angle 1tE6≥-0.900196 then ...
     node 9 else up
 7  if Angle N5<0.0199299 then node 10 elseif Angle N5≥0.0199299 then ...
     node 11 else down
 8  class = down
 9  class = up
10  if Angle N5<-0.681997 then node 12 elseif Angle N5≥-0.681997 then ...
    node 13 else down
11  class = down
12  class = down
13  class = up
```

Listing 5.1: Example for resulting If-then code for the tree presented in Figure 5.11

5.5 Summary of Results and Comparison between HMM and DT

Both, the HMM and the DT approach lead to similar results when tested on the same set of training and recognition data. The results for both approaches are summarized in Table 5.8. The recognition rate for the full gesture set of ten gestures is around 90% for both approaches, for eight gestures the recognition rate is around 95%. Finally, reducing the gesture set to a minimum of four gestures, resulted in a recognition rate of nearly 100%. With the larger gesture sets the DT shows a slightly better performance than the HMM. However, when the minimum gesture set is used for recognition the HMM delivers a bit higher accuracy than the DT. Additionally there has to be considered that due to stability problems the results of the HMM-based gesture recognition were achieved by using only one Gaussian mixture. This might be a reason for the poor recognition results for the gestures "rightdown" and "leftdown". This problem has to be ruled out in future work, and it has to be evaluated whether the HMM will then show better recognition results. Still, the implementation of a HMM-based embedded gesture recognition device using the UCOS sensor might be hard to realize because of the limited computing resources within the device.

[%]	**HMM10**	**DT10**	**HMM8**	**DT8**	**HMM4**	**DT4**
DO	96.15	90.38	94.23	96.15	100.00	98.88
UP	92.31	98.08	96.15	94.23	100.00	97.75
LE	96.15	98.08	96.15	100.00	100.00	100.00
RI	96.15	94.23	98.08	98.08	100.00	100.00
RU	76.92	90.38	92.31	96.15		
RD	76.92	90.38	94.23	96.15		
LU	98.08	96.15	92.31	96.15		
LD	92.31	88.46	94.23	94.23		
RF	90.38	86.54				
WV	80.77	78.85				
Avg. Rate	89.61	91.15	94.71	96.39	100.00	99.16

Table 5.8: Recognition results, comparison between HMM and DT for 10, 8 and 4 gestures

For both, the HMM approach and the DT approach, the results show that a reduced gesture set leads to more accurate recognition results than a full gesture set. Better overall recognition results might be achieved when only parts of the gestures are recognized and the decision about the meaning of a gesture is resembled by sequential combinations of gesture parts recognized by a second decision instance. It might also be useful to use all data produced by the UCOS sensor for gesture recognition, not only the output of the embedded tracking function.

Under this prerequisites, the decision tree seems to be the better solution for implementing the prototype of an embedded gesture recognition device, especially when using a reduced gesture set of eight gestures. However, both methods still need further improvements to increase the usability of such a system for gesture recognition.

CHAPTER 6

Conclusion and Future Work

In the last chapter of this thesis Section 6.1 is used to compare the results of this work with related works. In this section it is also concluded whether results accomplished by using UCOS-generated AE data for gesture recognition are promising for further implementations of a gesture recognition system based in this sensor.
Some ideas for further development and brief outlook on future tasks on UCOS-based gesture recognition is presented in Section 6.2.

6.1 Conclusion and Discussion of the Results

There are different scientific and commercial solutions for human gesture recognition. In this work gesture recognition was evaluated as a new application for an existing sensor generating address event data. Especially, the tracking function of the sensor which is intended to support the sensors's function of counting people, was used as the source of data. In order to get recognition results, a basic framework with Hidden Markov Models and Decision Trees was designed for training and recognition. Both of the machine learning approaches were trained with different gestures and gesture sets of different size in order to evaluate the general functionality of the approaches and to find out how many different gestures can be detected using the selected features. Both, the Hidden Markov Model approach and the Decision Tree approach, were evaluated within an execution of a leave-one-out cross validation. Simple feature calculations were used, because a future development should lead to the development of an embedded gesture recognition sensor based on the UCOS device. The calculation of the features still needs a lot of computational resources which is likely to cause a high latency between the execution of a gesture and its recognition in a future embedded solution. When previously calculated features are presented to the recognition algorithms especially the Decision Tree leads to good results. It is also a method that is simple to implement, making an early prototype based on this solution possible in near future. The HMM also leads to accurate recognition results although a higher number of mixtures of Gaussian could not be used. If these problems are ruled out in the future the HMM might show even better results than the DT.

Paper/Work	**Recog. of:**	**Sensor**	**Recog. Method**	**Nr. Gestures**	**Average Accuracy**
Kristensson 2012 [53]	dyn. hand gestures	Kinect	template matching	>10	92.70%
Lang 2011 [56]	german sign language	Kinect	HMM	10	97.70%
Ramamoorthya 2003 [76]	dyn. hand gestures	mono camera	HMM	5	97.00%
Wang 2008 [90]	body gestures	stereo camera	HMM	9	91.96%
Hahn 2011 [42]	dance motions	UCOS family	HMM	8	97.33%
This study	dyn. hand gestures	UCOS	HMM	10	89.61%
				8	94.71%
				4	100.00%
			DT	10	91.15%
				8	96.39%
				4	99.16%

Table 6.1: Comparison of the results of this study and related works

In Table 6.1 a comparison of this study and related works is provided. From this overview it is visible that the recognition accuracy of the methods used in this work are in a similar range as the results from related works. It is also visible, that the works where a lower number of different gestures have to be recognized provide higher recognition rates than the works where more different gestures have to be recognized. The same observation was made in this study. Basically, the results of this work show that it is possible for the UCOS sensor to be used as gesture recognition device. They also show that the outcome of this work is in a range with competing approaches for gesture recognition. Finally, these results build the base for further developments and improvements on UCOSbased gesture recognition.

6.2 Future Work

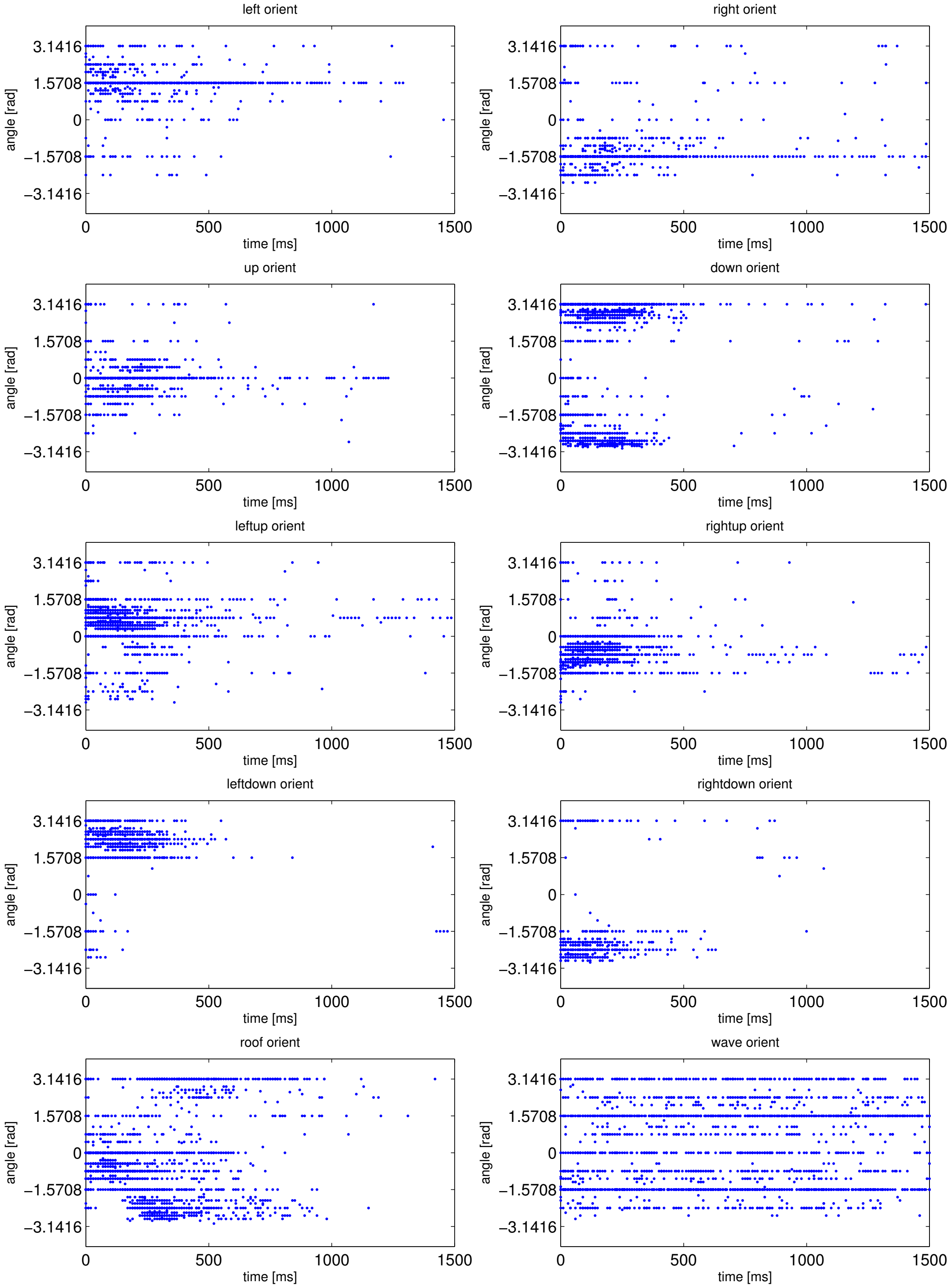
The major task for future work is to find and solve the problem within the HMM recognition which prevents using more than one Gaussian mixture for the training of the HMMs. Additional filtering and preprocessing for the input data has to be evaluated, as it might lead to better results. Therefore further filters could be useful to improve the recognition rates. For this evaluation only the data produced by the embedded tracking function of the UCOS sensor was used as data source. Extracting features from all AE data using not only the tracks might also lead to more accurate recognition results. It might also be interesting to use the approach and the presented in [53] on the AE-based trajectories.

To get a more representative result it is necessary to use the infrastructure developed an built for this work on an increased number of training and test data sets. It is also necessary to record data from more than one person since the way the people perform gesture differs from person to person.

For everyday use it is also necessary to find a way to recognize and rule out irregular gestures and unintentional movements, so that the do not trigger a reaction on a gesture controlled interface.

APPENDIX A

Appendix A: Feature Plots


left orient

right orient

up orient

down orient

leftup orient

rightup orient

leftdown orient

rightdown orient

roof orient

wave orient

Figure A.1: Comparison of the feature "orientation" for ten different gestures

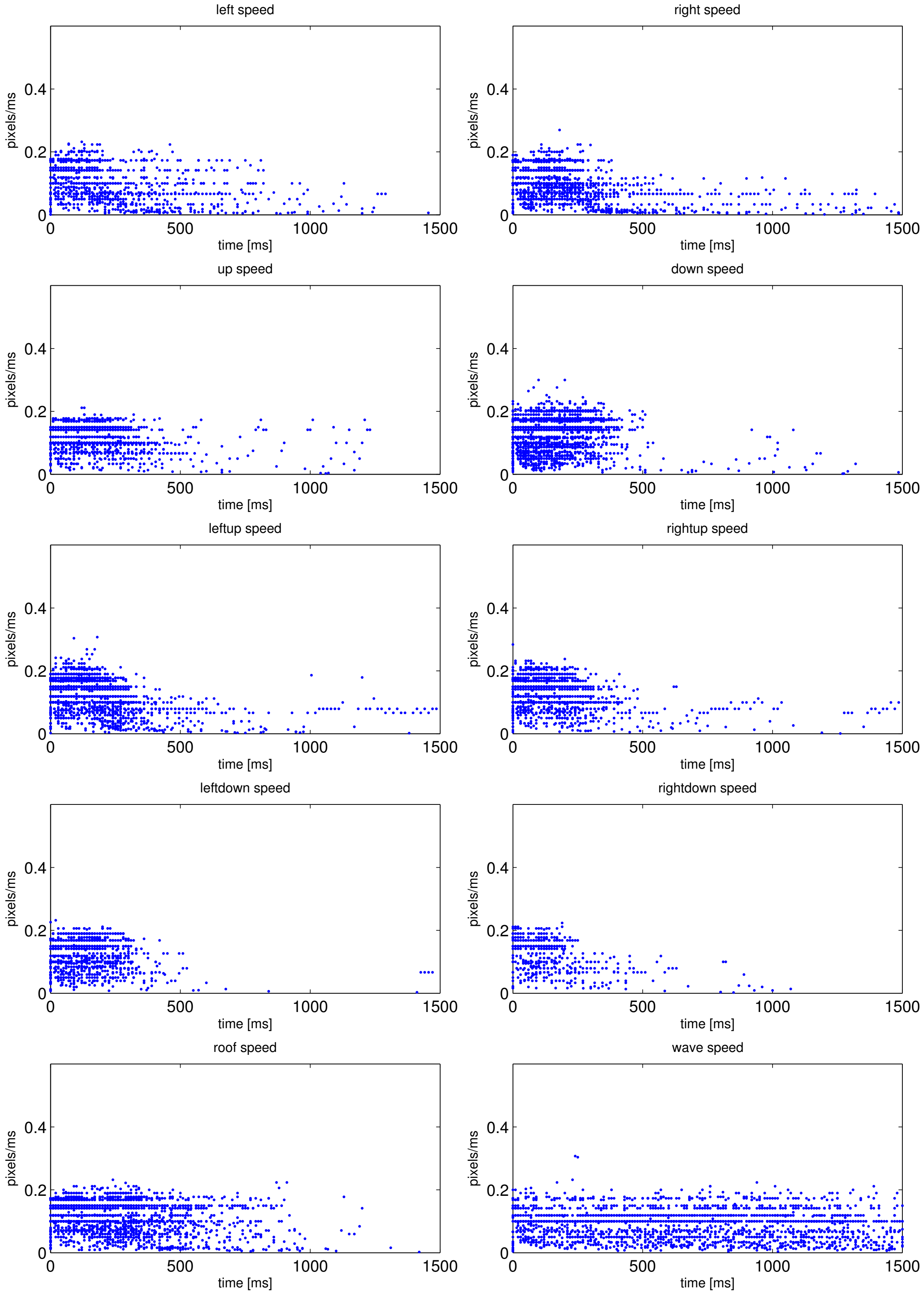


Figure A.2: Comparison of the feature “speed” for ten different gestures

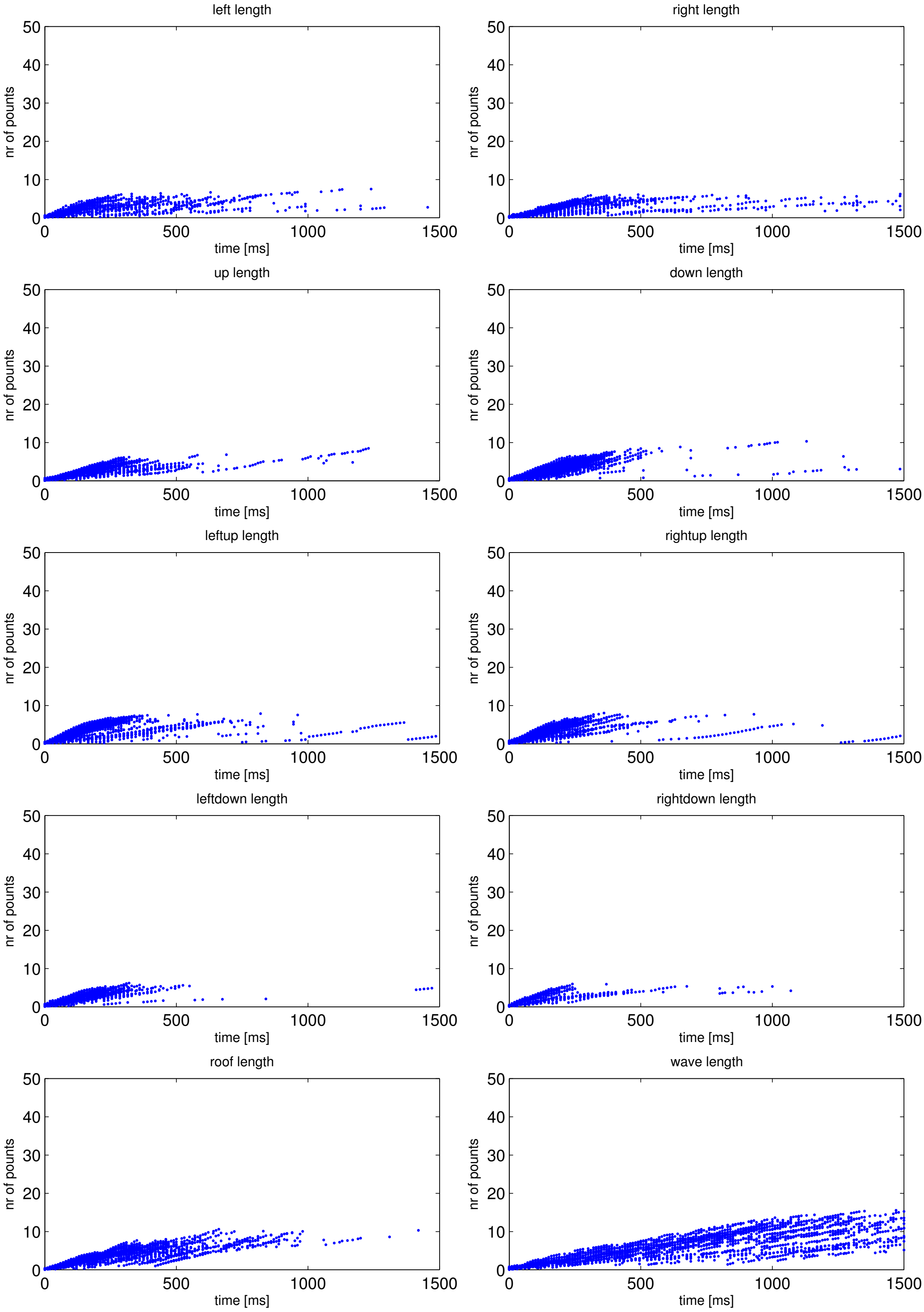


Figure A.3: Comparison of the feature “length” for ten different gestures

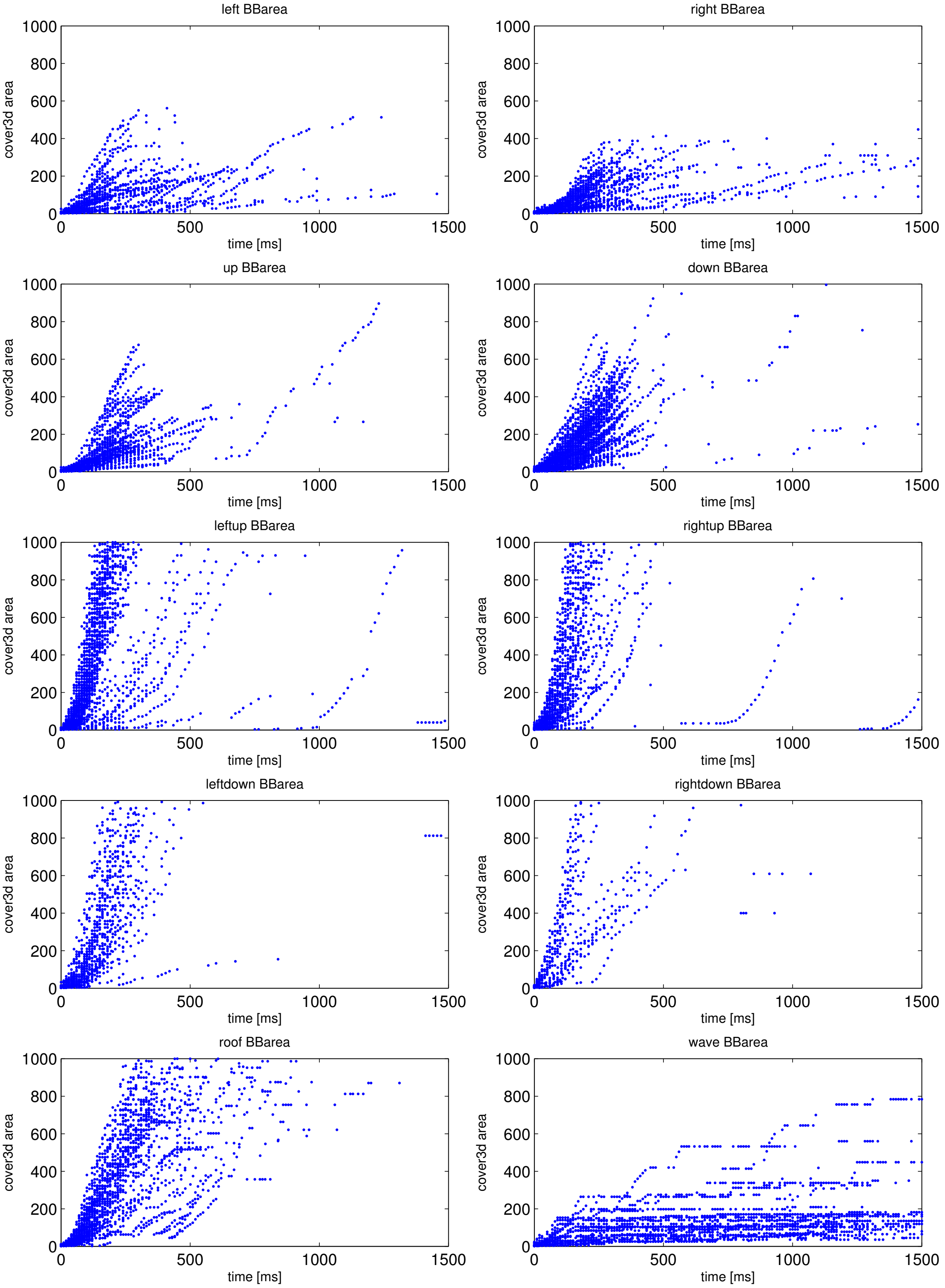


Figure A.4: Comparison of the feature “BB-area” for ten different gestures

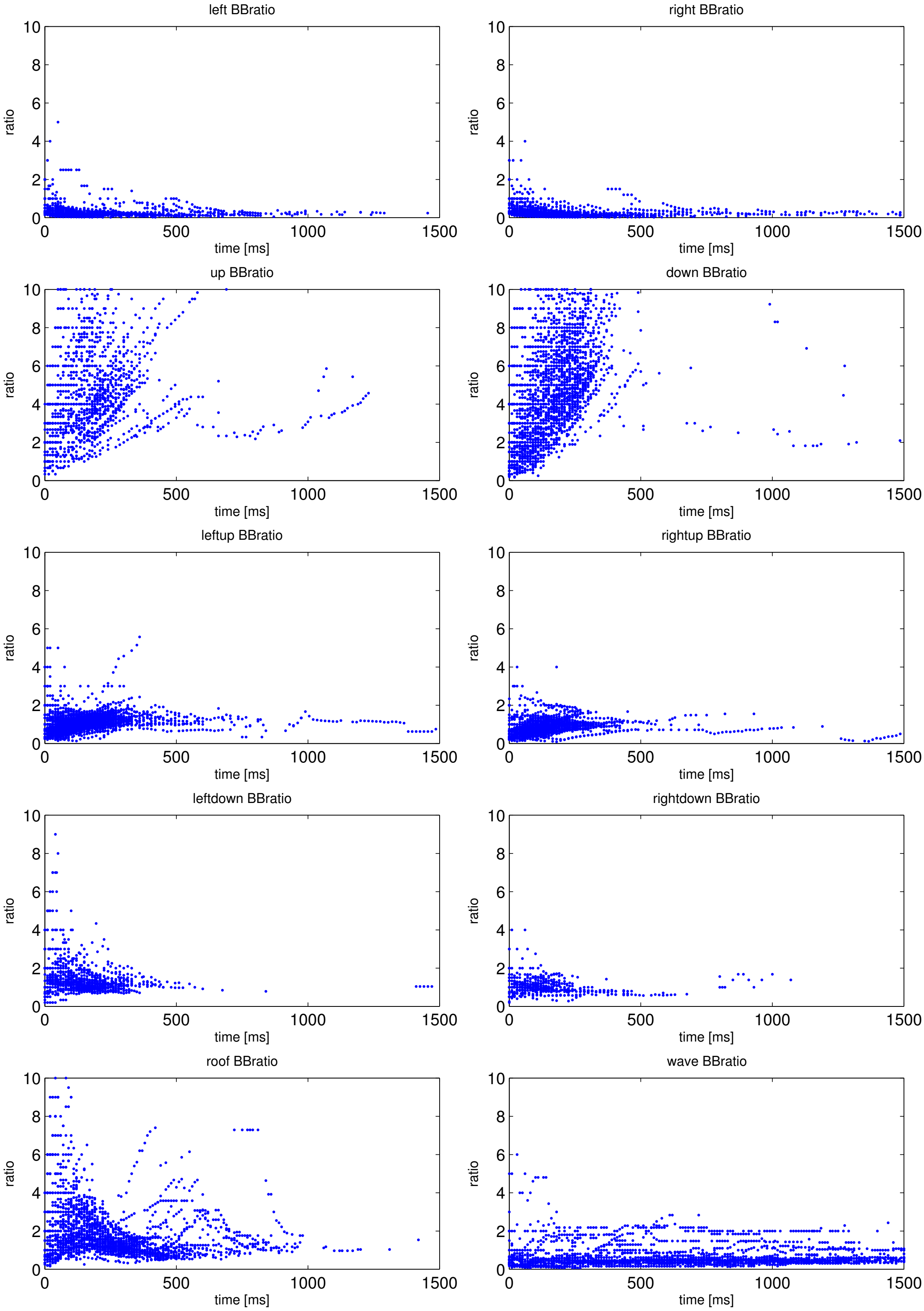


Figure A.5: Comparison of the feature "BB-ratio" for ten different gestures

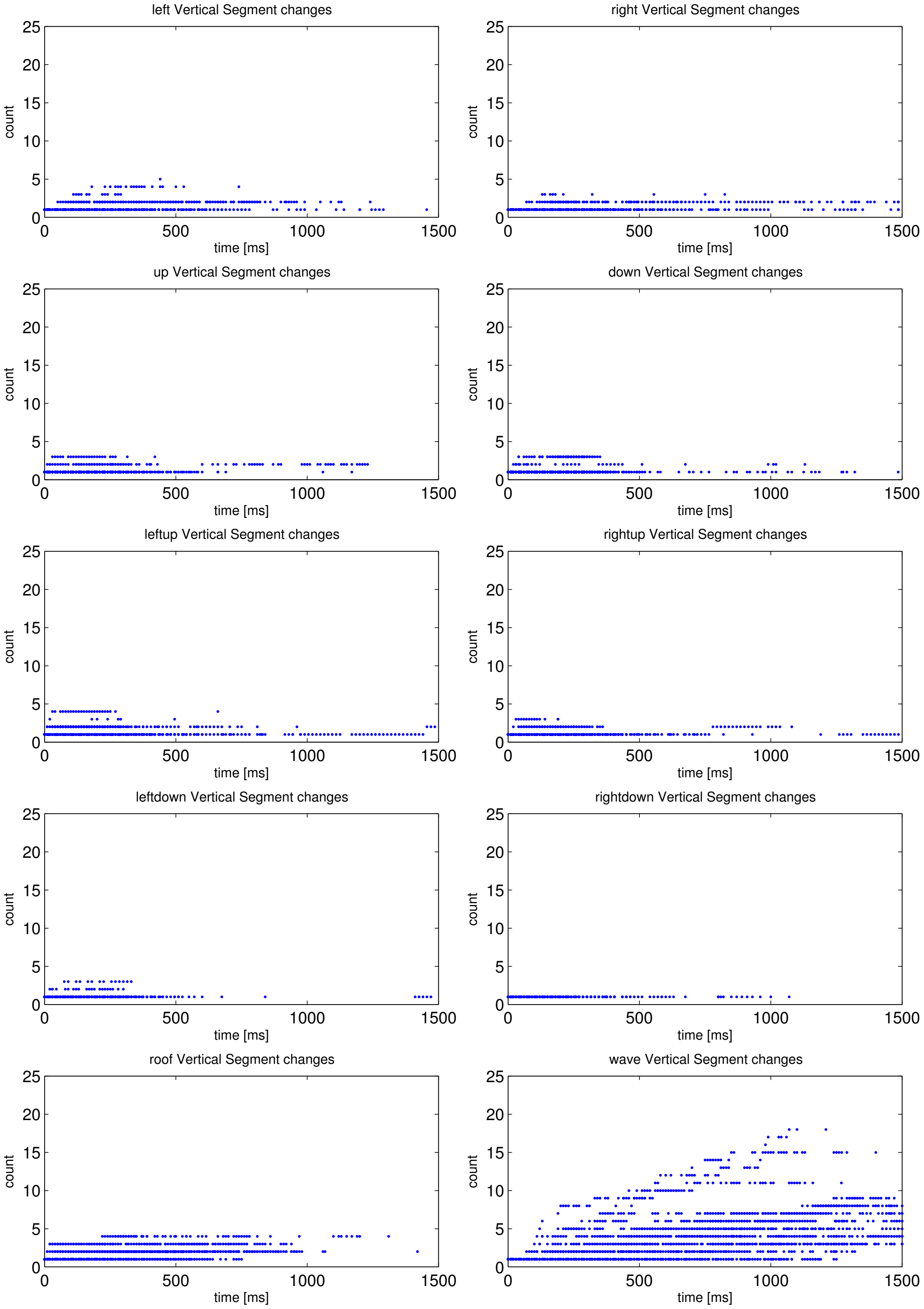
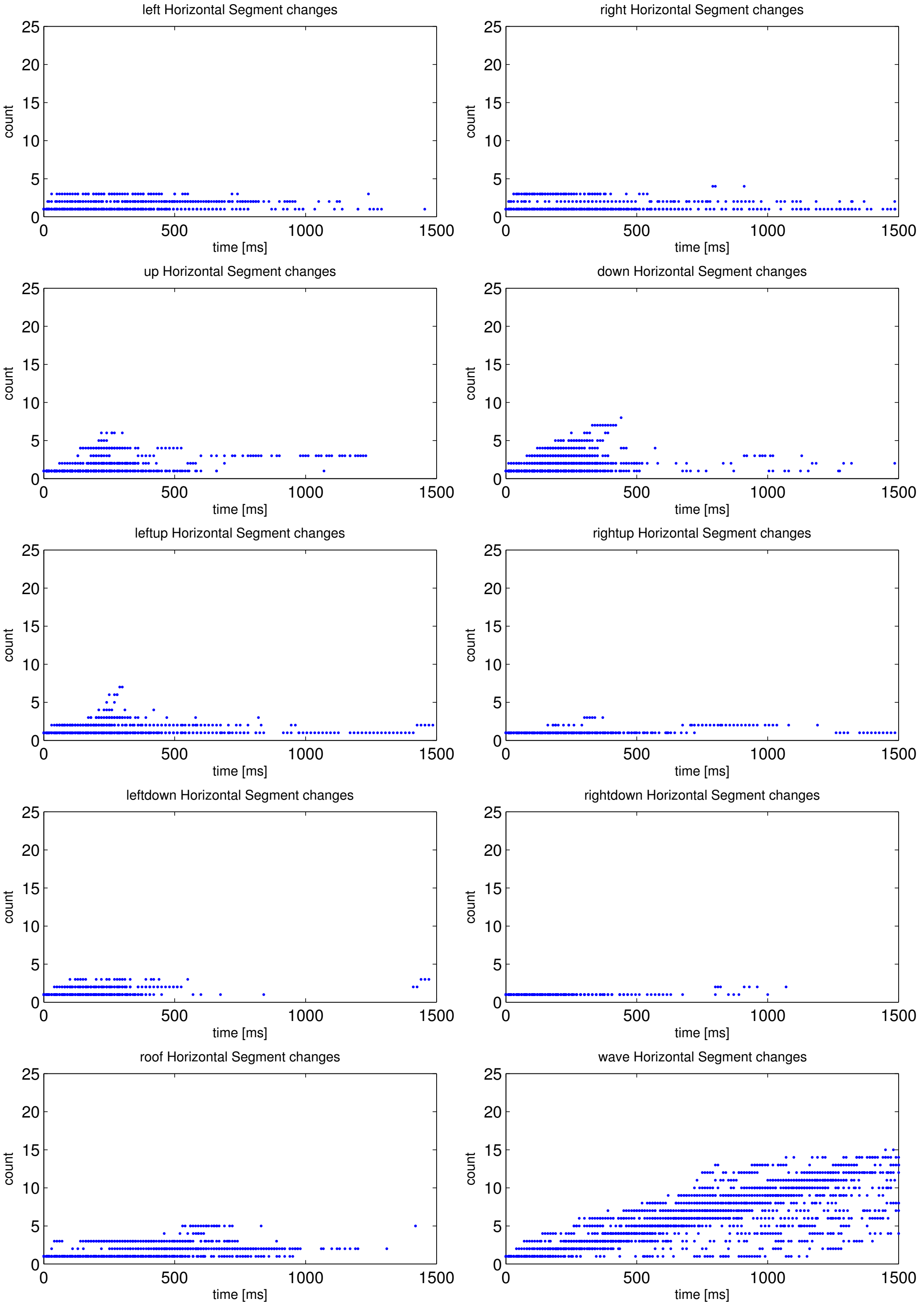


Figure A.6: Comparison of the feature “vertical direction changes” for ten different gestures



left Horizontal Segment changes

right Horizontal Segment changes

up Horizontal Segment changes

down Horizontal Segment changes

leftup Horizontal Segment changes

rightup Horizontal Segment changes

leftdown Horizontal Segment changes

rightdown Horizontal Segment changes

roof Horizontal Segment changes

wave Horizontal Segment changes

Figure A.7: Comparison of the feature “horziontal direction changeds” for ten different gestures

APPENDIX B

Appendix B: Code

B.1 Appendix B, Code: Filter different AER types

```
% bitmasks
%                             10987654321098765432109876543210
%                              3         2         1         0
% Stereo Evens (do nothing)
maskinfo(1) = uint32(bin2dec('00000000000000000000000000000000'));
% Scanlines
maskinfo(2) = uint32(bin2dec('00000010000000000000000000000000'));
% Counter numbers
maskinfo(3) = uint32(bin2dec('00000100000000000000000000000000'));
 % Tracks
maskinfo(4) = uint32(bin2dec('00000110000000000000000000000000'));
 % Crosses on beginning of tracks
maskinfo(5) = uint32(bin2dec('00001000000000000000000000000000'));

%check the status od the tree masking bits
mymask = uint32(bin2dec('00001110000000000000000000000000'));
%apply mask
index = find (bitand(ae(1,:),mymask)==maskinfo(type));

if (¬isempty(index));
    ae = ae(:,index);
else ae = [];
```

B.2 Appendix B, Code: HMM Chain

```
DB_Path = ...
'C:\Users\Matthias Zima\Documents\Informatik\Diplomarbeit\Data\tracks.mat';

gesture_names={'down' 'up' 'left' 'right' 'rightup' ...
               'rightdown' 'leftup' 'leftdown' 'roof' 'wave'};

%no diagonal gestures
%gesture_names={'down' 'up' 'left' 'right' 'wave'};

% Variables to Store LoglikV
Result.ids = [];
Result.gt =  {};
Result.loglikV_norm = [];

countup=1;

% Variables to save IDs of false deteced gestures
failindex=0;
false_ids=[];
% ************************************************************************/
% * Open Database
% ************************************************************************/
trackdata = load_trackdata_from_DB(DB_Path);

% ************************************************************************/
% * Select Tracks with specific quality from DB
% ************************************************************************/
% * Tracks with quality better or equal to 'tr_quality' are selected
% * for training and verification
% *     1 ... down      5 ... rightup       9 ... roof
% *     2 ... up        6 ... leftup        10... wave
% *     3 ... left      7 ... rightdown
% *     4 ... right     8 ... leftdown
% ************************************************************************/
if DB_select
    for ind=1:no_gestures
        track_ids{ind} = select_tracks( trackdata, gesture_names{ind}, ...
                                        tr_quality , '>');
        [¬,track_cnt(ind)] = size(track_ids{ind});
    end;

    % ********************************************************************/
    % * Take equal number of samples from all tracks (min) if selected
    % * so that all datasets have the same size. The number of tracks used
    % * depends on the tinyest gesture set. If there ar 150 up gesturere
    % * and 30 wave gestures recorded and annotated in the database, 30
    % * random  gestures of the other datasets will be picked.
    % * To reuse this dataset the used track_ids are sored in the output.
    % * HMM_chain can be started using previously stored track_ids.
    % ********************************************************************/
    if useequal
        sizedata = min(track_cnt);

        for ind=1:no_gestures
            rpointers = randperm(track_cnt(ind));

            pointers = sort(rpointers(1:sizedata));

            for index=1:sizedata
                ids_to_use{ind}(index) = track_ids{ind}(pointers(index));
            end;
        end;

        track_ids=ids_to_use;
    end;
end;

% ************************************************************************/
% * Get total number of tracks to be checked
% ************************************************************************/
countdown=0;
for index1=1:no_gestures
    [¬,temp]=size(track_ids{index1});
    countdown=countdown+temp;
end;

% ************************************************************************/
% * Initialize Confusion Matrix
% ************************************************************************/
Confusion=zeros(no_gestures);

% ************************************************************************/
% * keep each dataset once out of trainingdata, train with the rest and
% * validate
% ************************************************************************/
for index1=1:no_gestures
    [¬,no_tracks]=size(track_ids{index1});
    failindex = 0;
    % ********************************************************************/
    % * Generate Feature Cell Arrays for "fixed" gestures
    % ********************************************************************/
    for index3=1:no_gestures
        if index3≠index1
            FCA{index3}=getFCA(trackdata,track_ids{index3},featureparam);
        end
    end;
    disp('Fixed Feature Cell Arrays generated!')

    for index3=1:no_gestures
        if index3≠index1
            % ************************************************************/
            % * Train the HMMs with trainingdata
            % ************************************************************/
            HMM(index3)=train2(FCA{index3}.data_tr,Kg,Ns,nr_iter,ltr,Tresh);
        end;
    end;
    disp('Training of fixed HMMs finished');

    for index2=1:no_tracks
        % ****************************************************************/
        % * Show, how many tracks are left to be used.
        % ****************************************************************/
        text=['Processing, ', num2str(countdown), ' Tracks to go!'];
        disp(text);
        countdown=countdown-1;

        % ****************************************************************/
        % * Generate Feature Cell Arrays for "permutated" gestures
        % ****************************************************************/
        FCA{index1}=getFCA(trackdata,track_ids{index1}, ...
                    featureparam, track_ids{index1}(index2));

        disp('Permutated Feature Cell Array generated!')

        % ************************************************************/
        % * Train the permutated HMM with trainingdata
        % ************************************************************/
        HMM(index1)=train2(FCA{index1}.data_tr,Kg,Ns,nr_iter,ltr,Tresh);

        disp('Training of Permutated HMM finished');

        % ****************************************************************/
        % * Recognize gesture using test data
        % ****************************************************************/
        loglikV=ones(no_gestures,1)*-inf;
        [¬,FCA_len]=size(FCA{index1}.data_te);
        for index3=1:no_gestures
            [¬,loglikV(index3),¬] = recognize(FCA{index1}.data_te,...
                HMM(index3));
            %loglikV_orig(index3)=loglikV;

            Result.loglikV_norm(((index1-1)*index2)+index2,index3)=...
                loglikV(index3)/FCA_len;
            Result.loglikV_orig(((index1-1)*index2)+index2,index3)=...
                loglikV(index3);
        end
        %loglikV_norm=loglikV/FCA_len;


        Result.ids = [Result.ids ; FCA{index1}.id_te];
        Result.gt{countup,1} = gesture_names{index1};
        %Result.loglikV_norm(index2,index2) = ...
            [Result.loglikV_norm(index1) ; loglikV_norm];
        %Result.loglikV_orig = loglikV_orig;

        countup=countup+1;

        [¬,gesture_nr]=max(loglikV);

        % ****************************************************************/
        % * Store result in Confusion Matrix
        % ****************************************************************/
        Confusion(index1,gesture_nr)=Confusion(index1,gesture_nr)+1;

        % ****************************************************************/
        % * Store ID's of false classified tracks
        % ****************************************************************/
        if gesture_nr ≠ index1
            failindex=failindex+1;
            false_ids{index1}(failindex)=FCA{index1}.id_te;
        end

    end

end

% ************************************************************************/
% * store data for evalution
% ************************************************************************/
eval.hmmparam=hmmparam;
eval.trparam=trparam;
eval.featureparam=featureparam;
eval.Confusion=Confusion;
eval.false_ids=false_ids;
eval.track_ids=track_ids;
eval.FCA=FCA;
eval.Result=Result;

end
```

B.3 Appendix B, Code: DT Chain

```
DB_Path = ...
'C:\Users\Matthias Zima\Documents\Informatik\Diplomarbeit\Data\tracks.mat';

gesture_names={'down' 'up' 'left' 'right' 'rightup' ...
               'rightdown' 'leftup' 'leftdown' 'roof' 'wave'};

%no diagonal gestures
%gesture_names={'down' 'up' 'left' 'right' 'wave'};

% Variables to Store LoglikV
Data.F = [];
Data.gt =  {};
Data.fname =  {};
Data.track_ids1 = [];

countup=1;

% Variables to save IDs of false deteced gestures
failindex=0;
false_ids=[];
% **********************************************************************/
% * Open Database
% **********************************************************************/
trackdata = load_trackdata_from_DB(DB_Path);

% **********************************************************************/
% * Select Tracks with specific quality from DB
% **********************************************************************/
% * Tracks with quality better or equal to 'tr_quality' are selected
% * for training and verification
% *     1 ... down      5 ... rightup       9 ... roof
% *     2 ... up        6 ... leftup        10... wave
% *     3 ... left      7 ... rightdown
% *     4 ... right     8 ... leftdown
% **********************************************************************/
if DB_select
    for ind=1:trparam.no_gestures
        track_ids{ind} = select_tracks( trackdata, gesture_names{ind}, ...
                                        trparam.tr_quality , '>');
        [¬,track_cnt(ind)] = size(track_ids{ind});
    end;

    % ******************************************************************/
    % * Take equal number of samples from all tracks (min) if selected
    % * so that all datasets have the same size. The number of tracks used
    % * depends on the tinyest gesture set. If there ar 150 up gesturere
    % * and 30 wave gestures recorded and annotated in the database, 30
    % * random  gestures of the other datasets will be picked.
    % * To reuse this dataset the used track_ids are sored in the output.
    % * HMM_chain can be started using previously stored track_ids.
    % ******************************************************************/
    if useequal
        sizedata = min(track_cnt);

        for ind=1:trparam.no_gestures
            rpointers = randperm(track_cnt(ind));

            pointers = sort(rpointers(1:sizedata));

            for index=1:sizedata
                ids_to_use{ind}(index) = track_ids{ind}(pointers(index));
            end;
        end;

        track_ids=ids_to_use;
    end;
end;

% ******************************************************************/
% * Get total number of tracks to be checked
% ******************************************************************/
countdown=0;
for index1=1:trparam.no_gestures
    [¬,temp]=size(track_ids{index1});
    countdown=countdown+temp;
end;

% ******************************************************************/
% * Initialize Confusion Matrix
% ******************************************************************/
Confusion=zeros(trparam.no_gestures);
Confusion2=zeros(trparam.no_gestures);

% ******************************************************************/
% * keep each dataset once out of trainingdata, train with the rest and
% * validate
% ******************************************************************/
for index1=1:trparam.no_gestures
    % ******************************************************************/
    % * Generate Feature Matrix
    % ******************************************************************/
    Preds{index1}=getPredictors(trackdata,track_ids{index1},...
        featureparam, numsegments, gesture_names{index1});

    Data.F=[Data.F;Preds{1,index1}.F];
    Data.gt=[Data.gt;Preds{1,index1}.gt];
    Data.fname=Preds{index1}.fname;
    Data.track_ids1=[Data.track_ids1,track_ids{index1}];

end
Data.track_ids1=rot90(Data.track_ids1,3);
Data.track_id=track_ids;
```

```

[sizeDataset,¬]=size(Data.F);
countdown=sizeDataset;
good = 0;
bad = 0;

% ************************************************************************/
% * For Cross Validation: leave one out and train with the Rest
% ************************************************************************/
for index=1:sizeDataset
    msg=['Performing Crossvalidation, ',num2str(countdown) ,...
         ' Tracks to go!'];
    disp(msg)
    countdown=countdown-1;

    % select Testdata
    Test.F=Data.F(index,:);
    Test.gt=Data.gt(index,:);
    Test.track_ids1=Data.track_ids1(index,:);

    % leave out Testdata from trainingsdata
    storecnt=0;
    for ind=1:sizeDataset
        if ind≠index
            storecnt=storecnt+1;
            Train.F(storecnt,:)=Data.F(ind,:);
            Train.track_ids1(storecnt,1)=Data.track_ids1(ind,1);
            Train.gt(storecnt,1)=Data.gt(ind,1);
        end;
    end;

    tree = classregtree(Train.F, Train.gt, 'names', Data.fname);

    treecollection{index}=tree;

    disp('Tree generated')


    result=eval(tree,Test.F);

    disp('------------------------------');
    disp(Test.gt);
    disp(result);

    if strcmp(Test.gt,result)
        good=good+1;
    else
        bad=bad+1;
    end;

    perc=(good/(good+bad))*100;

    for index2=1:trparam.no_gestures
```

```
        if strcmp(gesture_names{index2},Test.gt)
            confpos_gt=index2;
        end;
        if strcmp(gesture_names{index2},result)
            confpos_rec=index2;
        end;

    end;

    Confusion(confpos_gt,confpos_rec)=Confusion(confpos_gt,confpos_rec)+1;
end;

% ************************************************************************/
% * store data for evalution
% ************************************************************************/
retvals.trparam=trparam;
retvals.featureparam=featureparam;

retvals.Data=Data;
retvals.Confusion=Confusion;
retvals.treecollection=treecollection;

retvals.perc=perc;

end
```

APPENDIX C

Appendix C: Sensor Settings

C.1 Settings of the UCOS Sensor

```
***************                 System   ***************
                Application Software: UCOS
 FLASH                    Board type: UCOS XL
 FLASH                     Sensor ID: 1068
 FLASH                      Location: Zim's_Desk
                             Version: 1.7FoSIBLE
                               Build: 1.7.6-3252r+
                                Date: 2011-11-14 11:49
                       Configuration: Standard
                           StereoLib: 2.18.1
                         MAC address: 00.60.36.07.20.64
                         Stereo/Mono: Stereo
                           CountMode: Statistic
                         Calibration: STOPPED
                   Calibration State: OK
 FLASH                     Uart Baud: 9600          (default: 9600)

***************                Network   ***************
 FLASH                    OutputMode: 1             (default: 1)
 FLASH                    StreamMode: 1             (default: 0)
                            Diagnose: 2             (default: 0)
                             BoardIP: 192.168.0.1
                              SendTo: 192.168.0.2
 FLASH                       ConPort: 20010         (default: 20010)
 FLASH                  SendToServer: 192.168.1.1
              Flash Memory Usage [%]: 99
 FLASH       Flash Sectors Available: 16            (default: 16, ...
     range=[1,16])
                       TFTP Transfer: no transfer
                         TFTP period: 10
```

```
                      Flash filename: <unknown>
 FLASH                Time server IP:
 FLASH   time synchronization period: 0             (default: 0)

***************            Application   ***************
 FLASH                         Focal: 4.000000      (default: 4.000000, ...
     range=[1.000000,36.000000])
 FLASH                        Height: 230           (default: 320, ...
     range=[100,1000])
 FLASH                        Shiftx: -1            (default: 0, ...
     range=[-20,20])
 FLASH                        Shifty: 2             (default: 0, ...
     range=[-40,40])
 FLASH                  DisparityMin: 1             (default: 4, ...
     range=[0,127])
 FLASH                  DisparityMax: 33            (default: 60, ...
     range=[0,127])
 FLASH                     FrameSize: 10            (default: 22, ...
     range=[8,64])
 FLASH                         AEMin: 4             (default: 4, ...
     range=[0,64])
 FLASH                          CMin: 0.100000      (default: 0.100000, ...
     range=[0.000000,1.000000])
 FLASH                        Blendl: 0             (default: 0, ...
     range=[0,100])
 FLASH                        Blendr: 0             (default: 0, ...
     range=[0,100])
 FLASH                      Blendtop: 0             (default: 0, ...
     range=[0,100])
 FLASH                      Blendbot: 0             (default: 0, ...
     range=[0,100])
 FLASH                      Gridsize: 16            (default: 32)
 FLASH                         Width: 100           (default: 300, ...
     range=[100,1000])
 FLASH                      Interval: 10            (default: 60, ...
     range=[0,86400])
 FLASH                         Pulse: 0             (default: 20)
 FLASH                    ActiveHigh: 0             (default: 1)
 FLASH                         Scl_1: 51            (default: 55, ...
     range=[0,127])
 FLASH                         Scl_2: 75            (default: 75, ...
     range=[0,127])
 FLASH                     Direction: 1             (default: 1)
 FLASH                  Nomatchstyle: 1             (default: 1)
 FLASH                  Stereooutput: 0             (default: 0)

***************                    DAC   ***************
 FLASH                     BiasCas 0: 4000000       (default: 4000000)
 FLASH                  BiasInjGnd 1: 1600000       (default: 1600000)
 FLASH                   BiasReqPd 2: 16777215      (default: 16777215)
 FLASH                  BiasReqPuX 3: 2100000       (default: 2100000)
 FLASH                 BiasDiffOff 4: 800           (default: 800)
 FLASH                     BiasReq 5: 23000         (default: 23000)
 FLASH                     BiasRefr 6: 0              (default: 0)
 FLASH                   BiasReqPuY 7: 9300000        (default: 9300000)
 FLASH                   BiasDiffOn 8: 600000         (default: 600000)
 FLASH                     BiasDiff 9: 10000          (default: 10000)
 FLASH                    BiasFoll 10: 3300000        (default: 3300000)
 FLASH                      BiasPr 11: 2000           (default: 2000)

***************                   Flags    ***************
                       Send to Server: no
                      Store in memory: no
                     Send over serial: no

***************             UDP Handler    ***************
                    connection status: DISABLED
 FLASH                      data type: XML packets
                          send period: 200
                         lost packets: 0
                       unsent packets: 0
 FLASH                           port: 20070          (default: 20070, ...
     range=[1000,65535])
OK
```

Listing C.1: UCOS settings for gesture recognition

Bibliography

[1] AAL association. ambient assisted living joint programme, access date: 14.August 2012. http://www.aal-europe.eu/.

[2] Allgemeines Gebärdenwörterbuch, access date: 05.October 2012. http://www.sign-lang.uni-hamburg.de/alex/index.html.

[3] CARE - safe private homes for elderly persons, access date: 14.August 2012. http://care-aal.eu/.

[4] FoSIBLE project homepage, access date: 01.May 2012. http://fosible.eu/.

[5] Kinect - project ideas, access date: 02. October 2012. http://openkinect.org/wiki/Project_ideas.

[6] Kinect for windows, access date: 02. October 2012. http://www.microsoft.com/en-us/kinectforwindows/.

[7] Nite middleware from primesense, access date: 05.October 2012. http://www.primesense.com/technology/nite3.

[8] Open NI, access date: 05.October 2012. http://openni.org.

[9] Sad - algorithm, access date: 05.October 2012.

[10] Samsung Smart TV, access date: 07. September 2012. http://www.samsung.com/at/tv-audio-video-home.

[11] SVM - Support Vector Machines , access date: 05.October 2012. http://www.support-vector-machines.org/.

[12] X-Box 360 + Kinect, access date: 07. September 2012. http://www.xbox.com/de-DE/Kinect.

[13] AIT, AIT Austrian Institute of Technology GmbH, Safety & Security Department, Donau-City-Straße 1, 1220 Vienna. *User manual: smart eye - UCOS Universal Counting Sensor*, version 1.5.0 edition, October 2010.

[14] D. Bass. Can kinect make windows cool again?, access date: 20. August 2012. http://www.businessweek.com/articles/2012-03-08/can-kinect-make-windows-cool-again.

[15] A.N. Belbachir, S. Schraml, A. Böttcher, M. Krenn, and M. Litzenberger. Stereo Vision Mittels einem Bio-inspirierten Stereo Sensor. In *Tagungsband zum 2. Deutschen AAL-Kongress in Berlin, Session 16.4*, pages 256–260, 2009.

[16] G. Bernstein, N. Lotocky, and D. Gallagher. Robot recognition of military gestures. Technical report, Cornell University - Department of Computer Science, 2011. CS 4758.

[17] A. Bharti. A decision tree approach to extract knowledge for improving satellite image classification. Master's thesis, Indian Institute of Remote Sensin, Institute of Remote Sensing, National Remote Sensing Agency (NRSA), Department of Space, Dehradun, India, 2004.

[18] M. Bhuiyan and R. Picking. Gesture-controlled user interfaces, what have we done and what's next? In *Proceedings of the Fifth Collaborative Research Symposium on Security, E-Learning, Internet and Networking (SEIN 2009)*, pages pp59–60., Darmstadt, Germany, November 2009.

[19] J.A. Bilmes. What HMMs and can do. In *IEICE TRANSACTIONS on Information and Systems*, volume E89-D, pages 869–891, 2006.

[20] J. Bobeth, S. Schmehl, E. Kruijff, S. Deutsch, and M. Tscheligi. Evaluating performance and acceptance of older adults using freehand gestures for tv menu control. In *Proceedings of EuroITV12*, pages 35–44, July 4-6 2012.

[21] B. Bongers. *Interaction with our electronic environment - an e-cological approach to physical interface design*. Hogeschool van Utrecht, Faculty of Journalism & Communication, 2004.

[22] C. Borgelt and R. Kruse. Attributauswahlmaße für die Induktion von Entscheidungsbäumen: Ein Überblick. In Gholamreza Nakheizadeh, editor, *Data Mining: Theorie und Anwendungen*, pages 77–98, Heidelberg, Germany, 1998.

[23] P.B. Braem. *Einführung in die Gebärdensprache und ihre Erforschung*. Internationale Arbeiten zur Gebärdensprache und Kommunikation Gehörloser. Signum, 1990.

[24] V. Brauer. Gestenerkennung mit einem Datenhandschuh. Paper 42, Forschungszentrum Arbeit und Technik (artec) Universität Bremen, 1996.

[25] P. Breuer. Entwicklung einer prototypischen Gestenerkennung in Echtzeit unter Verwendung einer IR-Tiefenkamera. Master's thesis, Universität Koblenz-Landau, Institut für Computervisualistik, Dezember 2005.

[26] P. Breuer, C. Eckes, and S. Müller. Hand gesture recognition with a novel IR time-of-flight range camera - a pilot study. In *MIRAGE'07 Proceedings of the 3rd international conference on Computer vision/computer graphics collaboration techniques*, pages 247–260, 2007.

[27] A. Camurri, B. Mazzarino, M. Ricchetti, R. Timmers, and G. Volpe. Multimodal analysis of expressive gesture in music and dance performances. In *Gesture-Based Communication in Human-Computer Interaction*, volume 2915 of *Lecture Notes in Computer Science*, pages 357–358, 2004.

[28] M. Caputo, K. Denker, B. Dums, and G. Umlauf. 3d hand gesture recognition based on sensor fusion of commodity hardware. In *Proceedings of Mensch & Computer 2012*, pages pages 293–302. Oldenbourg Verlag, 2012.

[29] B. Chakraborty, O. Rudovic, and J. Gonzàlez. View invariant human-body detection with extension to human action recognition using component-wise hmm of body parts. In *Proceedings of the 5th international conference on Articulated Motion and Deformable Objects*, AMDO '08, pages 208–217, 2008.

[30] A.N. Chielotam. Expressing indigenous knowledge through dance. In *African Journal of History and Culture*, pages 69–73. Theatre Arts Department, Nnamdi Azikiwe University, Awka, Anambra State, Nigeria, June 2012.

[31] C. Cortes and V. Vapnik. Support-vector networks. *Machine Lerning*, 20:pages 273–297, September 1995.

[32] Z. Cěrnekova, N. Nikolaidis, and I. Pitas. Single camera pointing gesture recognition using spatial features and support vector machines. In *Proceedings of the 15th European Signal Processing Conference (EUSIPCO 2007)*, pages 130–134, 2007. http://www.eurasip.org/Proceedings/Eusipco/Eusipco2007/Papers/A1L-F05.pdf, access date: 16. October 2012.

[33] P. Dondi, L. Lombardi, and M. Porta. Gesture recognition by data fusion of time-of-flight and color cameras. *World Acedemy of Science, Engineering and Technology*, 59:pages 1954–1960, November 2011. http://www.waset.org/journals/waset/v59/v59-367.pdf, access date: 16. October 2012.

[34] D. A. Norman; S. W. Draper. Cognitive engineering. In *User Centered System Design: New Perspectives on Human-Computer Interaction*, pages 31–61. Lawrence Erlbaum Associates, 1986.

[35] D. Droeschel, J. Stückler, and S. Behnke. Learning to interpret pointing gestures with a time-of-flight camera. In *Proceedings of the 6th international conference on Human-robot interaction*, HRI '11, pages 481–488, New York, NY, USA, 2011. ACM.

[36] R. Dugad and U. B. Desai. A tutorial on hidden markov models. Technical Report Technical Report No. : SPANN-96.1, Signal Processing and Artificial Neural Networks Laboratory Department of Electrical Engineering Indian Institute of Technology, Bombay Powai, Mumbai 400 076, India, May 1996.

[37] Yona Falinie, Abdul Gaus, and Farrah Wong. Hidden markov model - based gesture recognition with overlapping hand- head/hand-hand estimated using kalman filter. In *Proceedings of the 2012 Third International Conference on Intelligent Systems Modelling and Simulation*, ISMS '12, pages 262–267, Washington, DC, USA, 2012. IEEE Computer Society.

[38] F. Gebauer. Gestenbasierte Interfaces - Konzeption und Entwicklung unter Verwendung der Microsoft Kinect. Master's thesis, Fachhochschule Köln, Campus Gummersbach, 2012.

[39] E. Cartmill; S. Beilock; S. Goldin-Meadow. A word in the hand: action, gesture and mental representation in humans and non-human primates. In *Philosophical Transactions of the Royal Society B 367*, pages 129 – 143, 2012.

[40] H. Gunes, M. Piccardi, and T. Jan. Face and body gesture recognition for a vision-based multimodal analyzer. In *Proceedings of the Pan-Sydney area workshop on Visual information processing*, VIP '05, pages 19–28, Darlinghurst, Australia, Australia, 2004. Australian Computer Society, Inc.

[41] J. Guo. Hand gesture recognition and interaction with 3d stereo camera. Technical report, Department of Computer Science, Australian National University, November 2011.

[42] T. Hahn. Event-driven 3d vision for human actvity analysis in context of dance and fitness training of elderly people. Master's thesis, Vienna University of Technology, Faculty of Informatics, 2011.

[43] W. Husinsky. Skriptum zu: Grundlagen der Physik für InformatikerInnen, 2006.

[44] M.A. Isard. *Visual Motion Analysis by Probabilistic Propagation of Conditional Density*. PhD thesis, Robotics Research Group Department of Engineering Science University of Oxford, September 1998.

[45] O. Ivanciuc. Applications of support and vector machines in chemistry. In *Reviews in Computational Chemistry*, volume 23, pages 291–400, 2007.

[46] M.B. Kaâniche. *Human Gesture Recognition*. PhD thesis, Docteur en Sciences de l'Universite de Nice - Sophia Antipolis Specialite: Automatique, traitement de signal d'images, 2009.

[47] M. Karam. *A framework for research and design of gesture-based human computer interactions*. PhD thesis, University of Southampton, November 2006.

[48] F. Karray, M. Alemzadeh, J. Abou Saleh, and M.N. Arab. Human-computer interaction: Overview on state of the art. *Internattional Journal on Smart Sensing and Intelligent Systems*, 1:137–159, March 2008.

[49] D-I. Ko and G. Agarwal. Gesture recognition: Enabling natural interactions with electronics. Technical report, Texas Instruments, 2012. http://www.ti.com/lit/wp/spry199/spry199.pdf.

[50] R. Kohavi and F. Provost. Glossary of terms. *Machine Learning*, 30:271–274, 1998.

[51] E. Kollorz, J. Hornegger, and A. Barke. Gesture recognition with a time-of-flight camera. *International Journal of Intelligent Systems Technologies and Applications*, 5(3/4):334–343, November 2008.

[52] A. Kötteritzsch, S. Budweg, and M. Klauser. Förderung sozialer Interaktion durch Activity Communities für Senioren. In M. Ritter M. Eibl, editor, *Workshop-Proceedings der Tagung Mensch & Computer 2011*, pages 275–277. Universitätsverlag Chemnitz, 2011.

[53] O. Kristensson, T.F.W. Nicholson, and A. Quigley. Continuous recognition and of one-handed and two-handed and gestures using 3d and full-body motion and tracking sensors. In *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces*, IUI '12, pages 89–92, New York, NY, USA, 2012. ACM.

[54] P.O. Kristensson and L.C. Denby. Continuous recognition and visualization of pen strokes and touch-screen gestures. In *In Proceedings of Conference on Sketch-Based Interfaces & Modeling (SBIM)*, pages pages 95 –102., 2011.

[55] E. Kuhnke. *Körpersprache für Dummies*. Wiley-VCH Verlag GmbH & Co. KGaA, 1. auflage edition, August 2008.

[56] S. Lang. Sign language recognition with kinect. Master's thesis, Freie Universität Berlin - Institut für Informatik, 2011.

[57] C. Larsen. *Die zwölf Grade der Freiheit: Kunst und Wissenschaft menschlicher Bewegungskoordination*. Via Nova, 1995.

[58] H. Lee and A. Y. Ng. Spam deobfuscation using a hidden markov model. In *Proceedings of Second Conference on Email and Anti-Spam (CEAS 2005)*, 2005. also available on http://ceas.cc/2005 (access date: 19.08.2012).

[59] P. Lichtsteiner, C. Posch, and T. Delbruck. A 128x128 120db 15us latency asynchronous temporal contrast vision sensor. In *2006 IEEE International Conference on Solid-state Circuits (ISSCC)*, pages 508–509, February 2006.

[60] MathWorks. Statistics toolbox - classification, access date: 25.May 2012. http://www.mathworks.de/products/statistics/demos.html?file=/products/demos/ shipping/stats/classdemo.html.

[61] C.A. Mead. *Analog VLSI and Neural Systems*. Addison Wesley Publishing Company, 1989.

[62] S. Mitra and T. Acharya. Gesture recognition: A survey. In *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, volume 37, pages 311–324, May 2007.

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

[64] K. Murphy. Hidden markov model (hmm) toolbox for matlab, access date: 25.May 2012. http://www.cs.ubc.ca/ murphyk/Software/HMM/hmm.html.

[65] S. K. Murthy. Automatic construction of decision trees from data: A multi-disciplinary survey. *Data Mining and Knowledge Discovery*, 2:345–389, 1998.

[66] C.L. Nehaniv. Classifying types of gesture and inferring intent. In *AISB'05 Symposium on Robot Companions: Hard Problems and Open Challenges in Robot-Human Interaction*, pages 74–81, 2005.

[67] A. Nowakowska. Recognition of a vision approach for fall detection using a biologically inspired stereo vision sensor. Master's thesis, Vienna University of Technology, Faculty of Informatics, 2011.

[68] M. Pal and P. M. Mather. Decision tree based classification of remotely sensed data. In *Proceedings of the 22nd Asian Conference on Remote Sensing*, November 2001. http://a-a-r-s.org/acrs/proceeding/ACRS2001/Papers/DPA3-05.pdf, access date: 16. October 2012.

[69] Farid Parvini, Dennis McLeod, Cyrus Shahabi, Bahareh Navai, Baharak Zali, and Shahram Ghandeharizadeh. An approach to glove-based gesture recognition. In *Proceedings of the 13th International Conference on Human-Computer Interaction. Part II: Novel Interaction Methods and Techniques*, pages 236–245, Berlin, Heidelberg, 2009. Springer-Verlag.

[70] V. I. Pavlovic, R. Sharma, and T. S. Huang;. Visual interpretation and of hand and gestures and for human-computer and interaction: A and review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):677–695, July 1997.

[71] L. Pistilli, C. Eastwood, E. Wallach, L.V. Cleef, and A. Giuffre. The good, the bad and the ugly. DVD, January 1998. ASIN: 6304698798.

[72] F. Quek, D. McNeill, R. Bryll, S. Duncan, X.F. MA, C. Kirbas, K.E. McCullough, and R.Ansari. Multimodal human discourse: Gesture and speech. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 9(3):171–193, September 2002.

[73] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[74] J. R. Quinlan and R. Kohavi. C5.1.3 decision and tree discovery. Technical report, School of Computer Science and Engineering Samuels Building G08 , University of New South Wales, Sydney 2052 Australia, 1999.

[75] L.R. Rabiner. Readings in speech recognition. In Alex Waibel and Kai-Fu Lee, editors, *Readings in speech recognition*, chapter A tutorial on hidden Markov models and selected applications in speech recognition, pages 267–296. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[76] A. Ramamoorthya, N. Vaswania, S. Chaudhurya, and S. Banerjeeb. Recognition of dynamic hand gestures. In *Pattern Recognition - The Journal of the Pattern Recognition Society*, volume 36, pages 2069 – 2081, 2003.

[77] Z. Ren, J. Meng, J. Yuan, and Z. Zhang. Robust hand gesture recognition with kinect sensor. In *Proceedings of ACM Multimedia (MM'11)*, 2011.

[78] B. Resch. Mixtures of gaussians - a tutorial for the course computational intelligence, access date: 25.May 2012. http://www.spsc.tugraz.at/system/files/mixtgaussian.pdf.

[79] K. Roebuck. *Gesture Recognition*. Amazon Distribution GmbH, Leipzig, 2011.

[80] D. H. Rubine. *The Automatic Recognition of Gestures*. PhD thesis, Carnegie Mellon University, 1991.

[81] S. Sadek, A. Al-Hamadi, B. Michaelis, and U. Sayed. An action recognition scheme using fuzzy log-polar histogram and temporal self-similarity. *Hindawi Publishing Corporation - EURASIP Journal on Advances in Signal Processing*, 2011. http://asp.eurasipjournals.com/content/pdf/1687-6180-2011-540375.pdf, access date: 16. October 2012.

[82] H. Scheid. *Duden: Calculation and mathematics. Encyclopedia for school and practice. (Duden: Rechnen und Mathematik. Ein Lexikon für Schule und Praxis.)*. Dudenverl., Mannheim, 1994.

[83] S. Schraml, M. Zima, and M. Litzenberger. Formate für Address-Event Daten. Technical Report V18, AIT - Department Safety & Security Geschäftsfeld Neuroinformatics SNI, 2011. internal documentation about formats for AER data.

[84] G. Shen, C. Peng, Y. Li, C. Shi, Y. Zhang, and S. Lu. Dita: Enabling gesture-based human-device interaction using mobile phone. Technical report, Microsoft Research, 2009.

[85] N. Shokhirev. Hidden markov models by nikolai shokhirev, access date: 18. August 2012. http://www.shokhirev.com/nikolai/abc/alg/hmm/hmm.html.

[86] Silvergame. Serious online gaming plattform, access date: 14.August 2012. http://www.silvergame.eu/.

[87] Y. Song. *Multi-Signal Gesture Recognition Using Body and Hand and Poses*. PhD thesis, B.S. Computer Science and Engineering, Hanyang University, September 2010.

[88] C. Stergiou and D. Siganos. Neuronal networks, access date: 07.October 2012. http://www.doc.ic.ac.uk/˜nd/surprise_96/journal/vol4/cs11/report.html.

[89] A.D. Trunk, J.G. Kemeny, B. Babbitt, P.E. Haggerty, C. Lewis, P.A. Marks, F. Jensen, C.B. Marrett, L. McBride, H.C. McPherson, R.W. Peterson, T.H. Pigford, and T.B. Taylor. Report of the presidents commission on the accident at three mile island. Technical report, The Presidents Commission On The Accident At Three Mile Island, 1979.

[90] Y. Wang, T. Yu, L. Shi, and Z. Li. Using human body gestures as inputs for gaming via depth analysis. In *Proceedings of IEEE International Conference on Multimedia & Expo 2008*, pages 993–996, 2008.

[91] Wikipedia. Hand-gesture A-OK, access date: 07. September 2012. http://en.wikipedia.org/wiki/A-ok.

[92] J.O. Wobbrock, A.D. Wilson, and Y. Li. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*, UIST '07, pages 159–168, New York, NY, USA, 2007. ACM.

[93] J. Yamato, J. Ohya, and K. Ishi. Recognizing human action in time-sequential images using hidden markov model. In *IEEE*, pages 379–385. IEEE, 1992.

[94] Matthias Zima. TracksortGUI zur Annotation von UCOS generierten AE-Trajektorien. Technical report, TU Wien, Fakultät für Informatik, September 2012.