Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

Airwriting Recognition using Wearable Motion Sensors

Diplomarbeit am Cognitive Systems Lab
Prof. Dr.-Ing. Tanja Schultz
Fakultät für Informatik
Universität Karlsruhe (TH)

von

cand. inform.
Christoph Amma

Betreuer:
Prof. Dr.-Ing. Tanja Schultz
Dipl.-Inform. Dirk Gehrig

Tag der Anmeldung: 19. Februar 2009
Tag der Abgabe: 19. August 2009

Cognitive Systems Lab

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Karlsruhe, den 19. August 2009

Zusammenfassung

In der vorliegenden Diplomarbeit stelle ich ein neuartiges Eingabegerät für *Wearable Computing* Anwendungen vor. Mithilfe des entwickelten Systems kann handschriftlicher Text eingegeben werden, indem man in die Luft schreibt, als würde man auf eine imaginäre Tafel schreiben. Die nötige Sensorik kann in die Kleidung des Nutzers integriert werden. Dies ermöglicht eine Interaktion mit mobilen Computersystemen ohne die Notwendigkeit spezifische Geräte in der Hand zu halten. Zu diesem Zweck habe ich einen Datenhandschuh mit integrierten Inertialsensoren entwickelt. Der Handschuh ist mit drei orthogonalen Beschleunigungs- und drei orthogonalen Drehratensensoren ausgestattet, welche sich auf dem Handrücken befinden. Die erfassten Sensordaten werden über Funk an einen Rechner übermittelt. Der Handschuh verfügt hierfür über einen Mikrocontroller und eine Bluetooth Schnittstelle. Die vorliegende Arbeit beschreibt unter anderem den Hardwareaufbau, die Softwareimplementierung und das verwendete Kommunikationsprotokoll des Datenhandschuhs.

Zur Klassifizierung der Signale kommen Hidden Markov Modelle zum Einsatz. Ich untersuche in dieser Arbeit die Erkennung einzelner Zeichen sowie einzelner Worte. Dabei werden die Hidden Markov Modelle für ganze Worte durch Verkettung der Modelle für die Einzelzeichen gebildet. Ich habe Experimente zur Erkennung der Zahlen von 0 bis 9, der Buchstaben von A bis Z, sowie zur Erkennung englischer Worte aus einem eingeschränkten Vokabular durchgeführt. Für die Experimente zur Buchstabenerkennung wurden Daten von zehn Probanden aufgenommen und die Leistung des Systems für den schreiberabhängigen und für den schreiberunabhängigen Fall evaluiert. Die Ergebnisse der Experimente werden analysiert und die Hauptgründe für fehlerhafte Erkennung identifiziert, welche im Wesentlichen in der Ähnlichkeit einzelner Buchstaben und Schreibvarianten verschiedener Schreiber liegen. Die mittlere Erkennungsrate im schreiberabhängigen Fall beträgt 95.9%, im schreiberunabhängigen Fall 81.9%. Auf der Basis der Daten eines einzelnen Probanden und eines Vokabulars von 100 Worten wurden erste Experimente zur Erkennung ganzer Worte durchgeführt. Hier wurde im schreiberabhängigen Fall eine Erkennungsrate von 96% erreicht und im schreiberunabhängigen Fall eine Erkennungsrate von 82%. Ich habe ein Demonstrationssystem zur Worterkennung implementiert, welches die praktische Anwendbarkeit des entwickelten Eingabesystems belegt.

Abstract

In my work, I introduce a wearable computing device for recognition of text written in the air, like on an imaginary blackboard. For that purpose, I designed and implemented a data glove, based on inertial sensors. The data glove is equipped with three orthogonal gyroscopes and three orthogonal accelerometers to measure hand motion. Data is processed and sent to a computer via Bluetooth. Based on the signals the glove delivers, I developed an HMM based recognizer, using the existing Janus Recognition Toolkit. Several experiments were performed to optimize and evaluate the system. Specifically, experiments were performed on single digit, single character, and word recognition. For the task of character recognition, ten test persons contributed to the data collection. Writer-dependent, as well as writer-independent recognition was evaluated and arising problems were analyzed in detail. The writer-dependent recognition rate on single character recognition was 95.3% on average. The average rate of writer-independent recognition was 81.9%. Based on a small vocabulary of 100 words, first experiments in word recognition were conducted and recognition rates of 96% were reached for the writer-dependent case and 82% for the writer-independent case. Finally, a real-time demonstration system was implemented to show the functionality of the system in practice. While there has already been some research in the field of airwriting recognition, using wireless sensor-equipped pens instead of a data glove, to the best of my knowledge, this is the first work on recognizing whole words written in the air.

Contents

1. Introduction

Hand gestures have always been part of human communication. Gestures are used in non verbal speech, when acoustic communication is not applicable or not desirable, for example, over large distances or in noisy environments. Gestures are also used as an additional information channel in verbal communication. Thus, it is obvious to use gestures as an interface for human-computer interaction. For that reason, computers have to be able to recognize these body movements. Various advances have been made in this direction. One can identify two basic approaches to the task of machine based gesture recognition, namely external and internal systems, which are distinguished by the sensor set-up used. External systems, like almost all vision based solutions, rely on cameras, installed in the environment of the user. Internal systems incorporate body-worn sensors. As external systems are unobtrusive, since the user does not have to carry any special devices, internal systems are independent of a special environment, like a sensor equipped room. They are even suitable for outdoor use, which is hard to accomplish with vision based systems. Body worn systems normally utilize modern MEMS[1] inertial sensors to detect motion. Commonly used sensors are accelerometers, gyroscopes and magnetometers. The body-worn sensor approach is extensively studied in the context of wearable computing and mixed reality applications. For such systems, the question of an ideal input device is still open. While pressing keys on a small keyboard does not seem to be the optimal method, hand gesture recognition might permit the design of an intuitive input device. For this purpose, any kind of predefined gesture set could be used, but it would be appealing to use characters and digits, which can be seen as special gestures, since users would not have to learn a new gesture system.

In [BCK+03], [OCB+04] and [KCK06] the idea of a new gesture based interface for wearable computing is introduced. It consists of an inertial sensor-equipped pen-like wireless device, enabling the user to write in the air, like on an imaginary blackboard. The written text is recognized and can be further processed. I call this type of input "airwriting". Experiments on recognition, up to the single character level, show promising results. Assuming, such a device would work reliably on whole sentences, it would offer great possibilities. It could be integrated into an ubiquitious

[1]micromachined electro-mechanical system

computing device like a mobile phone or PDA. There would be no need to integrate a touch screen for a tablet PC like input method and as result, the device itself could be very small. The type of input would perfectly complement speech recognition, since both interfaces have complementary properties. Gesture or airwriting input works well in some situations where speech input may not. It is insusceptible to acoustic noise, unobstrusive in quiet areas and, while not providing perfect privacy, does offer more of it than speech. But such a pen like device still needs to be held in hand, and thereby, has to be taken out of a pocket or bag and put back there. In an augmented reality working scenario, users might hold tools in their hand, which they don't want to put aside to make a small note. In such a situation, a solution that does not rely on an auxiliary device, but is attached directly to the body, without restricting the user, would be desirable. A good option would be the use of a data glove, ideally as lean and slim as possible.

Consequently, in this work, I introduce a wireless data glove for airwriting recognition. The device is equipped with acceleration and angular rate sensors to measure hand movement with six degrees of freedom. I implement a recognizer for 3D-space handwriting up to word level. The functionality is shown in experiments and by means of a writer-dependent real time demonstration system. The possibilities of writer-independent systems are also examined.

1.1 Objectives

The objective of this work is two-fold. The first part is the design and implementation of a wireless data glove based on an inertial sensor to measure hand motion and the appropriate software device driver to record the sensor signals. The second part is the design, implementation and evaluation of a handwriting recognition system to be used in combination with the data glove.

First, appropriate solutions for the sensing, on-device data processing, wireless transmission and power supply were found. Based on the chosen components, the circuit design was done and afterwards implemented. The device contains a microcontroller which was programmed in order to implement the required functionality. On the computer side, a device driver was implemented and integrated into a software system to receive, process and save the sensor data.

Second, an appropriate design of a recognizer was chosen and implemented. An existing speech recognition toolkit was used to build the recognizer. Experiments were planned and executed to optimize and test the recognizer. Based on the achieved results, more challenging tasks towards continuous airwriting recognition were tackled. Arising problems were analyzed and, if feasible, possible solutions are proposed.

The result of the work offers a good insight into the possibilities of the device and its use in airwriting recognition. A real time demonstration system was implemented to work as proof of concept for the overall task.

1.2 Structure of this Work

This work is structured in six chapters, of which this introduction is the first.

In the second chapter, the theoretical background to understand this work is explained. I expect the reader to have some background knowledge in computer science, especially in the field of machine learning and statistical modeling. I give a brief introduction to Hidden Markov Models and how they can be used to build recognizers for problems like handwriting recognition. I also introduce inertial sensors and their characteristics. Finally, I give an overview of related work in this field of research.

In the third chapter I will describe the design and implementation of the data glove up to the implementation of the device driver for the data receiving system. This will not be a comprehensive documentation, but a functional overview of the system.

In the fourth chapter I investigate the problem of handwriting recognition with an inertial sensor based data glove in a general way. I will especially compare it to the task of conventional on-line handwriting recognition and the special problems that arise in our case.

The fifth chapter contains the detailed description of all performed experiments, including the experimental set-ups and the results. The results are analyzed, problems are identified and possible solutions are discussed. The chapter ends with the introduction of a real time demonstration system.

In the sixth and last chapter, I give conclusions on the work done and will suggest reasonable future work possibilities, based on the achieved results.

2. Background

In this chapter, I give the necessary theoretical background information to understand this work, as well as an overview of related work in this research field. The chapter is structured in three main parts. First, I give an overview over MEMS inertial sensors. An outline of their properties and signal characteristics is given, as well as a short introduction to the theory of inertial navigation systems. This section intends to give the reader a feeling for what is special about inertial sensors, what can be done with them and what problems typically arise, when working with them. Second, I introduce Hidden Markov Models, which form the theoretical basis for the implemented recognizers in this work. The whole theory is only considered under the special application of handwriting recognition. Third and last, I give an overview of related research work.

2.1 Inertial Sensors

In this section I will give a short introduction to inertial sensors, especially MEMS type gyroscopes and accelerometers. I will outline the basic functional principles, as well as resulting signal properties and limitations. The concept of an inertial navigation system will also be explained.

2.1.1 Accelerometers and Gyroscopes

Both Accelerometers and Gyroscopes belong to the category of inertial sensors. They have been used for decades, mainly for navigation purposes. While there exist different technologies from early mechanical systems to laser based sensors, I will make use of *micromachined electromechanical system* (MEMS) sensors. These types of sensors are based on silicon and the same techniques as in integrated circuits production can be used to produce them. Thereby, the sensors are small, cheap, robust and leightweight. The performance is good enough for gesture recognition, especially handwriting. Accelerometers measure the linear acceleration of the sensor device in one axis. Gyroscopes measure the angular velocity around one axis. It should be emphasized, that gravitational force is always present and therefore, acting on an accelerometer. As result, every accelerometer measures a part of the earth acceleration, depending on its orientation relative to the direction of the gravitational force.

2.1.2 Inertial Navigation

If one uses three accelerometers and gyroscopes, it is theoretically possible to reconstruct the performed 3D-space trajectory from sensor signals. In my work, I do not incorporate the estimation of the trajectory, but since I sometimes refer to the absence of trajectory information as a problem, I explain the estimation process and arising problems in short. When using inertial sensors for motion recognition, it would be practical, if we could reconstruct the 3D-space trajectory, the sensor performed. One needs a so called strapdown inertial measurement system, which consists of three orthogonal oriented accelerometers and three orthogonal gyroscopes, to do so. Given the starting position, attitude and velocity, the position can be tracked. The angular velocity, given by the gyroscopes, must be integrated once to compute the attitude and with this information, the acceleration can be cleaned of the gravitational force. The acceleration must then be integrated twice to get the actual position. This has to be done every timestep. The system can only track its position relative to the starting position and since there is no possibility to get information about the absolute position in space, any errors in position tracking will cummulate. Experiments of Woodman in [Woo07] show that the simple application of the equations of an inertial navigation system does not give feasible results. The accumulated error, caused by sensor drift, is too big. In [BCK$^+$03] and in [OCB$^+$04] more advanced methods for trajectory reconstruction in the special case of handwriting are introduced, but even with these methods reconstruction ist only possible for single characters or short words. The estimation of the trajectory is not mandatory, motion recognition can also be done on the raw sensor signals, in this case acceleration data and angular velocity data. In this work, I do not incorporate trajectory estimation for the recognition task, I only use acceleration and angular velocity data.

2.2 Hidden Markov Models

Hidden Markov Models (HMM) are a widely used technique for sequence modelling. They belong to the category of statistical models, in contrast to deterministic models. In deterministic models, some kind of specific knowledge of the underlying process is used to model it, whereas in a statistical model, the underlying process is assumed to be a stochastic process, which can be parametrized in some way. HMMs model such a process by estimating the internal parameters based on example data. Of course by using a statistical model, it is assumed that the underlying process is a *parametric* random process. In case of Hidden Markov Models, the process is assumed to be a Markov Process. I give a short introduction to the theory of Hidden Markov Models along the excellent tutorial by Lawrence R. Rabiner [Rab89].

2.2.1 Discrete Markov Processes

In general, a stochastic process is in a distinct state at a distinct time, where the future state sequence is uncertain due to some kind of indeterminacy. I will speak of discrete processes as discrete in time and state space. That means, the state space is finite and the system undergoes changes from state to state, only at distinct and equally spaced times. Let's denote the states as $S = \{S_1, S_2, \dots, S_N\}$, where N is the total number of states. The points in time will be denoted as $t = 1, 2, \cdots T$ with

T denoting the duration of the process. The actual state at time t is denoted by q_t and the whole state sequence by Q.

A Markov Process is a special stochastic process. In a First-order Markov Process, the Probability for a state transition depends only on the actual state, instead of all past states, as in a general stochastic process. Also this probability is independent of time, i.e. the state transition probability for two states does not depend on the actual point in time, at which this transition is performed. These two assumptions are called the Markov Property. We can write the state transition probabilities a_{ij} as

$$a_{ij} = P(q_t = S_j | q_{t-1} = S_i), \qquad 1 \leq i, j \leq N \tag{2.1}$$

The probabilities for leaving a state must sum up to one, that means, the equation

$$\sum_{j=1}^{N} a_{ij} = 1, \qquad 1 \leq i \leq N \tag{2.2}$$

must hold.

We can now write all transition probabilities as a Matrix $A = \{a_{ij}\}$. The initial state distribution is denoted by $\pi = \{\pi_i\}$ with

$$\pi_i = P(q_1 = S_i), \qquad 1 \leq i \leq N \ . \tag{2.3}$$

In a Hidden Markov Model, the state is not directly observable, thus hidden. The observable output of a state is a probabilistic function of the state, so one does not know, in which state the model is, only the output of the current state is known. The output sequence is denoted by $O_1, O_2, \ldots, O_T$ and the discrete output alphabet by $V = .\{v_1, v_2, \cdots, v_M\}$, with M beeing the number of possible output values. In a typical application, for example handwriting recognition, an HMM would exist for every character, consisting of a distinct number of states related to different parts of the shape of a character. These internal states would not be known, neither would they be observable. But the output of the states would be the character, for example as points in the x,y plane or in our case acceleration and angular rate values. The output itself is a propability function, it can be a mapping to a discrete output space or a continuous probability density function. In our case, we use continuous density probability functions, which are typically approximated by Gaussian mixture models. The observation probability distribution in state j will be denoted by $b_j(x)$ with $B = \{b_j\}$ being the vector of all distributions. The tuple (A, B, π, N) gives a complete description of the system and is called a Hidden Markov Model. In general, HMMs can be fully connected, which means, state transition can occur from every states to every other state, including self loops. But by setting some transition propabilities to zero, a different topology can be imposed on the model. Figure 2.1 shows an HMM with a left-right topology, i.e. every state has exactly one predecessor and exactly one successor state, except the self loop. The start and stop states are special, since they do not have a predecessor or successor respectively. We can fully describe an HMM of this form by the state transition matrix and the output propability density functions. We denote it by

$$\lambda = (A, B) \quad . \tag{2.4}$$

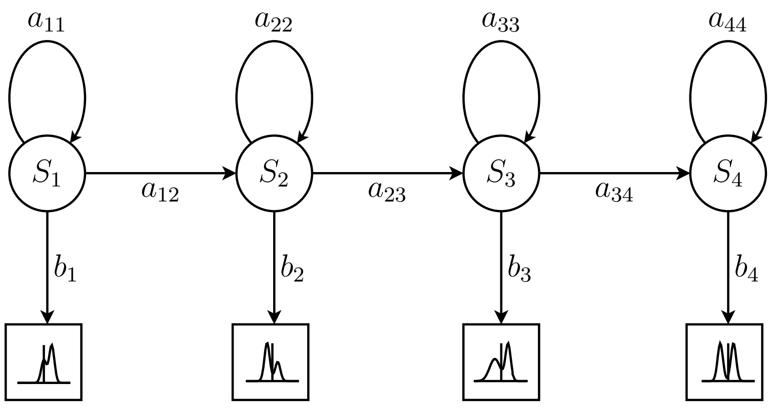


Figure 2.1: Schematic HMM with four states and a left-right topology

2.2.2 The three basic Problems

We can now formulate the three basic problems for HMMs:

1. The Evaluation Problem: Given a model λ and the output sequence $O = O_1, O_2, \dots, O_T$, compute the probability $P(O|\lambda)$ of the output sequence.
2. The Decoding Problem: Given a model λ and the output sequence $O = O_1, O_2, \dots, O_T$, find the optimal state sequence for the output sequence, which is normally done by maximizing $P(Q|O, \lambda)$.
3. The Training Problem: Given a model λ and the output sequence $O = O_1, O_2, \dots, O_T$, how do we adjust the model parameters (A, B) to ensure that $P(O|\lambda') > P(O|\lambda)$?

Let's suppose we have a solution to all three problems, how can we use HMMs to build a pattern recognizer for some kind of signal? The preprocessed sampled signal corresponds to the output sequence O. Concernig the task of handwriting recognition we might have such signals for each of the characters of the alphabet. Then we are able to *train* a model for each character by computing the solution to problem 3, which incorporates the solution to problem 2, since the parameters (A, B) are adjusted such that $P(O|\lambda)$ is maximized. If we then get a new signal without knowing the character to which it belongs, we can compute the probability to produce this specific output signal for each of the character models by using the solution to problem 1. In case of the alphabet, this would result in 26 probabilities, since we have one model per character in the alphabet. Our hypothesis will be the character, whose model gives the highest probability. I will give a very brief overview of the algorithms to solve these problems. This should give the reader an idea how they work, for a detailed discussion, I again refer to [Rab89].

The Evaluation Problem

We can compute the probability of a model λ for a given observation sequence O by simply summing up the individual probabilities of every possible state sequence.

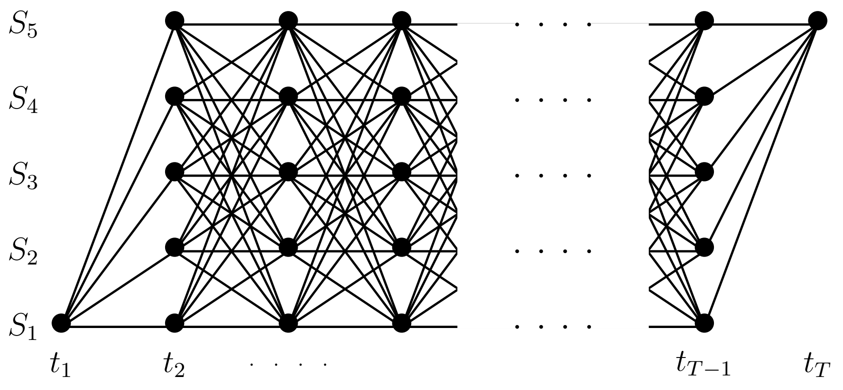


Figure 2.2: Complete state graph with all possible transitions for a 5 state HMM

Let's denote one such state sequence by $Q = q_1, q_2, \ldots, q_T$. We can now write the probability of the observation sequence for the model as

$$P(O|\lambda) = \sum_{\text{all } Q} P(O|Q,\lambda)P(Q|\lambda) \tag{2.5}$$

$$= \sum_{q_1,q_2,\ldots,q_T} \pi_{q_1} b_{q_1}(O_1) a_{q_1 q_2} b_{q_2}(O_2) \cdots a_{q_{T-1} q_T} b_{q_T}(O_T). \tag{2.6}$$

The direct calculation would be too expensive in terms of computation, as it would have exponential complexity in the length of the observation. In figure 2.2 the graph of all possible state sequences over time is drawn. Based on the graph, the task can be reformulated to compute the probabilities of all paths through the trellis structure from left to right. We can take advantage of the fact, that at every time step all paths remerge into the same N state nodes of the graph. The individual path probabilities can be computed by induction over time using dynamic progamming. Starting with the first state, the probability for all successor states, given the observation, can be computed. This is done time step by time step until the final state is reached. This algorithm is called Forward-Algorithm, because the trellis graph is traversed in "forward" direction to compute path probabilities.

The Decoding Problem

To solve problem 2, namely finding the optimal state sequence for a given model and output, we first have to define an optimality criterium. In our case, that would be the state sequence with the highest probability for the observation sequence. Formally this means maximizing the probability $P(Q|O,\lambda)$ or equivalently $P(Q,O|\lambda)$. Thinking of the trellis structure in figure 2.2, the task is to find the single path from left to right, which maximizes this probability. This can be done similar to the solution of the evaluation problem. Instead of summing up the probabilities for all paths, only the maximum probability for reaching a state is taken into account and used for the next induction step. Thereby, the maximum single path probability can be computed and by keeping track of the used path, the best state sequence is given. This algorithm is called the Viterbi algorithm.

The Training Problem

In training all parameters of the model are adjusted to maximize the probability $P(O|\lambda)$. The parameters are the transition probabilities A and the parameters of the mixture models B. This is normally done using the Baum-Welch algorithm, relying on the forward-backward algorithm, which can be used to compute individual transition probabilities for a given timestep and observation. I will use a faster alternative based on the Viterbi algorithm, which is a simplification of the forward-backward algorithm. Another simplification is not to train the state transition probabilities at all, thus initialize them with constant values. I make use of these simplified HMM training procedure, since application to speech recognition showed, that this has no significant impact on the performance of the recognizer. The Viterbi training algorithm has two phases. First the optimal state sequence is computed using the Viterbi algorithm introduced in the last paragraph. In the second phase the parameters of the Gaussian mixtures are reestimated based on this path through the trellis structure. By iterating over all training samples, this is done for every training example sequence.

2.3 Related Work

The field of gesture recognition has been extensively studied in the past. Two main approaches can be identified, vision based systems and systems using body-mounted sensors, including hand held devices. The latter type of system has the advantage of beeing self-contained, while vision based systems normally rely on external cameras. External dependencies are not desirable for our device and thus vision based systems are not considered any further.

For the purpose of human-computer-interaction, hand gestures were proposed as an intuitive possibility. Research was done on recognition, as well as on comprehension of gestures in general and in the context of coverbal articulation in speech [Sow08]. Also the speciality of interfaces without visual feedback was studied [OP07]. Concerning the task of handwriting recognition in the air, one can identify different research areas in gesture recognition with similarities to our task.

The recognition of sign language deals with an unsegmented stream of gestures, from which a recognizer has to extract words and sentences. Basically for handwriting recognition the same has to be done, but typically in sign language recognition systems, different and more sensors are used, since also finger movements must be taken into account.

The other research area of interest is gesture recognition with the use of inertial sensors, which gained a lot of attention in the recent years. Here, the focus lies on spotting and recognizing gestures, typically done with a small device. Most of the work is limited to the recognition of isolated gestures, instead of sequences.

Sensors and Devices

Different sensing techniques were used for gesture recognition with body-worn sensors. Brashear et al. [BSLJ03] use a hybrid system, consisting of a head mounted camera and accelerometers worn at the wrist for sign language recognition. While such a hybrid system seems promising for sign language recognition, it would be too cumbersome for us. Cheok et al. [CGKP02] give an overview of accelerometer based devices for wearable computer applications, including an accelerometer equipped pen. Kim et al. [KCK06] use a wireless pen device, equipped with accelerometers and gyroscopes for airwriting character recognition, reaching good recognition results. This can be seen as a proof of concept for our intended application. Often, the sensors are integrated into a device, which has to be held in hand and therefore is not optimal in sense of unobtrusitivity. Instead, data gloves have been proposed for gesture recognition, for example by Hofmann et al. and Kim et al. [HHH98], [KC01]. These gloves usually deliver a lot of information including finger movement and position. For that reason, the gloves are equipped with many sensors all over the hand, which makes them quite cumbersome. They also often contain a magnetic sensor for global position determination. These sensors depend on an externally induced magnetic field, which makes them unattractive for wearable interfaces, since they only work inside the induced magnetic field. Alternatively, an accelerometer equipped bracelet for gesture recognition was proposed by Hein et al. [HHK09] and Amft et al. [AAS+09] use accelerometers included in a watch to control it by gestures. In this work, I try to combine the advantages of data gloves with the convenience of small hand held devices by designing a very slim and unobstrusive glove based interface.

Pattern Recognition

A widely used technique in gesture recognition are HMMs, due to their ability to classifiy temporal signal sequences. Hofmann et al. [HHH98] use HMMs for accelerometer based gesture recognition with a data glove. They evaluate the performance of ergodic and left-right models and can find no significant difference for the task of gesture recognition. Schlömer et al. [SPHB08] implement an HMM based recognizer to distinguish five gestures. Amft et al. [AAS+09] integrate the HMM decoder directly in their gesture controllable watch with good recognition results. They also deal with the problem of spotting gestures in continuous sensor data by using a threshold HMM model. Liang et al. [LO98] and McGuire et al. [MHRS+04] make use of HMMs for continuous sign language recognition. In this case HMMs are especially useful, since they can be easily concatenated to construct more complex models from gesture primitives. This is also done on stroke level by Kim and Chien [KC01] for single gesture recognition, achieving high recognition rates. HMMs have been applied with success in traditional handwriting recognition, Plamondon and Srihari [PS00] give a survey on this extensively studied field of research. Choi et al. [CLL06] use an accelerometer equipped pen to recognize single digits by writing on a horizontal surface. They evaluate different methods like HMMs and Dynamic Time Warping. In their case DTW performs better than HMMs. Kim et al [KCK06] use Bayesian Networks, which are a generalization of HMMs, to recognize characters written in the air.

Other pattern recognition techniques have also been used. Hein et al. [HHK09] implement a C4.5 decision tree based recognizer directly on the device. Decision trees are used to lower computational needs. Rehm et al. [RBA08] compare Naive Bayes, Multilayer Perceptrons and Nearest Neighbour classifiers for gesture recognition, including single digit recognition. They achieve very high recognition rates, also for the user independent case. Wu et al. [WPZ+09] use Support Vector Machines on frames to distinguish twelve single gestures. Bang et al. [BCK+03] introduce a pen-type device for writing in the air, together with a method for reconstructing the trajectory without performing any recognition. Cho et al. and Oh et al.

[CK04],[OCB$^+$04] use this device for single digit recognition, using Bayesian Networks and Fisher Discriminant Analysis. They used the estimated trajectory as well as the raw sensor signals as features. The combination of both features gave the best results, but recognition on the raw sensor signals also reached high result and performed better, than recognition on the trajectory alone. In numbers, the best recognizer on raw sensor signals reached a recognition ratio of 93.43% and the combination of raw sensor signals together with the estimated trajectory reached 95.04%.

All these works on inertial sensor based gesture, digit and character recognition have in common, that isolated gestures are tried to be recognized. It seems that few attempts have been made to recognize continuous gestures. By continuous gestures, I mean sequences of basic gestures. The set of gestures is also normally limited to a small set of 10 to 20 uni-stroke gestures, which are often chosen and defined for ease of discrimination. For sign language recognition, complex models for continuous recognition have been developed, but since sign language includes finger movements, normally other sensing techniques or hybrid systems are used. While methods like Support Vector Machines or Nearest Neighbour classifiers might perform well for the case of single gesture recognition, it is hard to apply these methods to recognize continuous gestures, which is needed for continuous handwriting recognition.

Few works exist with the aim of handwriting recognition on base of accelerometer signals. In [ZDLK08] Zhou et al. use a pen device with inertial sensors for single character recognition. They use frequency space features in conjunction with Self Organizing Maps for the classification. In their experiments, the writing is performed on a horizontal surface. In [KCK06] Kim et al. tackle the problem of handwritten character recognition in the air. They introduce a ligature model based on Bayesian Networks to estimate pen-up and pen-down strokes for the task of airwriting recognition with a pen-like device. By that, traditional handwriting training data can be used to train the recognizer. High recognition rates are reached for digit and character recognition.

Up to my knowledge there have not been any further publications on the specific matter of handwriting in the air. Especially no extensive work on writer-independent recognition and recognition of whole words seems to exist. In my work, I examine writer-independency in detail. Besides the requirement to write in block capital letters, the test persons were free to write as usual. This leads to a lot of differing writing styles of certain characters and the effect on the recognition performance is analyzed. One left-hander contributed to the data set, which allows to evaluate a system trained on right-hander data on a left-handed person.

The recognition of whole words was not yet faced for the task of writing in the air. No work seems to tackle the question, how character gestures can be combined to recognize whole words. In my work, I do a first step in this direction by introducing a small vocabulary writer-dependent word recognizer using HMMs to model the motion between consecutive characters.

3. An Inertial Sensor Data Glove

In this chapter, I introduce a sensor glove equipped with three accelerometers and three gyroscopes. Data is sent to a computer via Bluetooth. I will outline the motivation for developing such a device, the concept and the implementation.

3.1 Problem Analysis

As we want a mobile wearable input device, we can formulate requirements such a system should meet:

- hands-free operation, no additional devices have to be held
- unobtrusive, ideally one should not even notice wearing a computer device
- robust in sense of radio link reliability and mechanic influence
- reasonable runtime without recharging batteries
- independent of environmental influences like temperature

Hands-free operation is an important point here. Following the vision of a computer that is only perceived when needed and does not interrupt other activities executed by the human user, it clearly is not desireable to fetch some kind of device out of your pocket to use your computer. Thus the system should truly be worn by the user. In addition, the system should be unobtrusive, at best imperceptibly. On the technical side, the system must be accurate enough for the specific task of handwriting recognition. Experiments showed that this is not a problem, even though we are not going to reconstruct the 3D trajectory. MEMS sensors are inherently temperature dependent, which is a problem, since the device is intended to be used in different surroundings, especially also outdoors. So temperature dependency must be faced, but fortunately, temperature calibrated products already exist. The sensor should also be leightweight, small and power efficient. The data link to the data processing computer must be reliable.

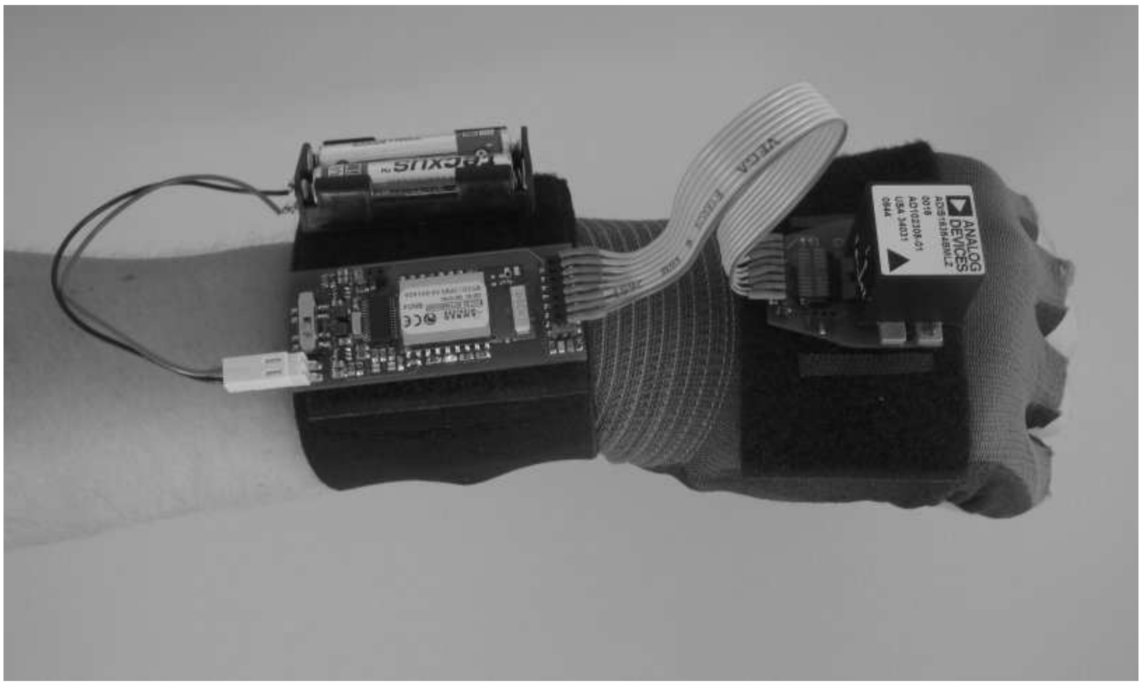


Figure 3.1: The data glove with all parts worn by a user.

3.2 Design

In this section I introduce a design of a sensor glove as input device, which aims to satisfy the above criteria. I will first give an overview of the design and afterwards describe the functional components in more detail. For a detailed technical description, I refer the reader to the data sheets.

3.2.1 Overview

The device consists of four basic functional components, the sensor, a microcontroller, a Bluetooth chip and the power supply. Figure 3.1 shows the developed data glove. A lot of different radio technologies exist and choosing one is always a trade-off. I have chosen Bluetooth, because it is an industry standard and there are highly integrated solutions available, so programming effort is low. Also, it is already integrated in almost all modern laptops or PDAs, thus there is no extra hardware needed on the computer side. It is also robust to a certain degree against interferences from other radio devices. On the other side, it is not the most power efficient and space saving technology.

The basic functional principle of the device is rather simple. The microcontroller reads data from the sensor and sends it to the Bluetooth chip, which sends it over an established Bluetooth link to a computer. As sensor, I took an Analog Devices ADIS16364 inertial measurement unit. The microcontroller is a Texas Instruments MSP430 with 16Mhz and 8K of flash memory. As Bluetooth Interface, I use the Amber Wireless Bluenicecom3 module. Figure 3.2 shows the block diagram of the system. The microcontroller communicates with the sensor over a SPI[1] interface. The communication with the Bluetooth module is done over a standard UART[2]

[1]SPI = Serial Peripheral Interface
[2]UART = Universal Asynchronous Receiver Transmitter

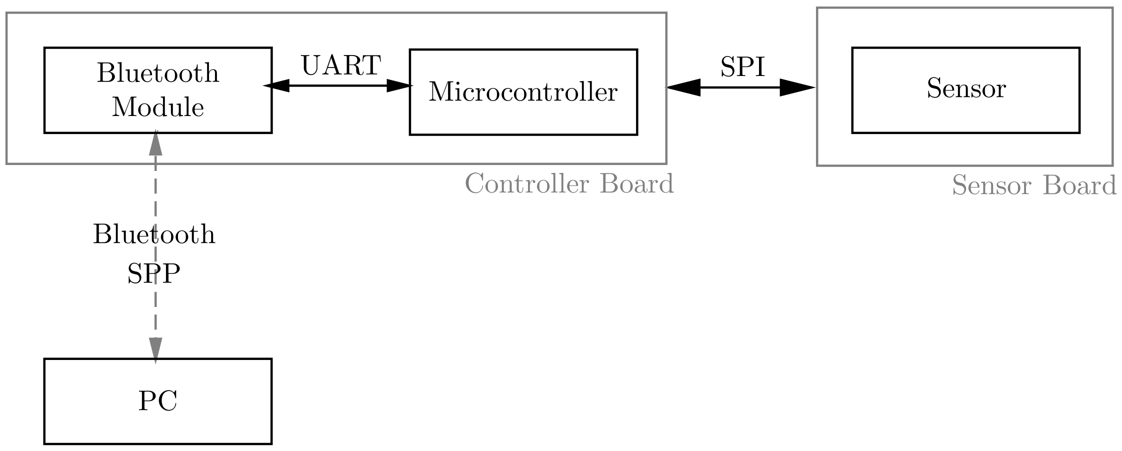


Figure 3.2: Device components shown in a block diagram with connecting communication protocols.

serial interface. The device has two modes of operation, send mode and command mode. In send mode, the microcontroller simply sends all data from the sensor over the Bluetooth link to the computer. In command mode, instructions can be sent to the device over Bluetooth. For example, there are commands to read battery charge status or to write to the sensor control registers. In command mode no measurement data is sent and in send mode, no commands can be sent, except the command to leave send mode. The Bluetooth module operates in slave mode, that means, when powered up, it will wait for connection requests. It will accept any request and establish the connection using the Serial Port Profile (SPP). This Profile is a Bluetooth standard and represents a point-to-point serial connection. From the software side, the connection appears as standard serial connection.

3.2.2 The Sensor

Analog Devices offers ready to use inertial measurement units. They all contain 3 orthogonal accelerometers and 3 orthogonal gyroscopes. They are already factory calibrated and some models do have an internal temperature compensation. The whole sensor comes in a package of size 2.3 cm × 2.3 cm × 2.3 cm. Figure 3.3 shows the sensor already mounted on the interface board. I'm using an ADIS16364[3], which offers temperature calibration from −20 °C to 70 °C. The sensor has an internal sampling rate of 819.2Hz, which can also be set to lower values. The internal accelerometers have a measurement range from $-5g$ to $5g$, the gyroscopes can be set to three ranges, namely ±75 °/s, ±150 °/s or ±300 °/s. The digital output values have 14bit, thus resolution of the accelerometers is 1 mg and for the gyroscopes 0.05 °/s for the ±300 °/s setting respectively. Therefore every value needs two bytes. Considering only one channel and taking the maximum data rate of the sensor, we get $2 \cdot 8 \cdot 819.2\,\text{bit/s} = 13\,107.2\,\text{bit/s}$ as maximal needed data rate per channel. We have six channels of measurement data. To ensure, all measurements from the sensor are sent and received, there will be an additional counter channel. For every sample a counter will be incremented by one and will be sent over the Bluetooth link with the associated data sample. Thereby the application is able to check, if all succesive

[3]Datasheet Analog Devices ADIS16364, revision B, www.analog.com

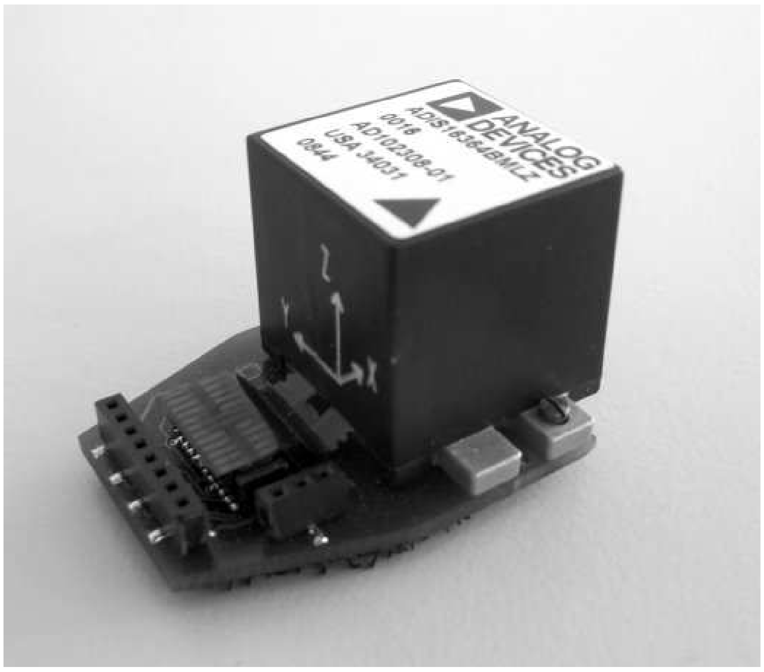


Figure 3.3: The Sensor board.

data packets are received. I will use 2 bytes for the counter, so there are 7 data channels in total. The total data rate sums up to 91 750.4 bit/s.

The sensor has an internal low-pass filter, which is enabled by default, but can be disabled. I will not use this filter, since filtering can as well be done on the application side and the raw sensor data offers more possibilities for using different filters. All functions and settings of the sensor are accessibly through memory mapped registers.

3.2.3 Microcontroller

As microcontroller, a Texas Instruments MSP430F2132 is used. It can operate at up to 16Mhz, has 8KB Flash memory and 512 byte RAM. It offers IO ports, which already can operate as SPI or serial interface and it has a 10bit analog digital converter, which will be used to measure battery voltage. The MSP is power efficient and offers different sleep and low power modes. Figure 3.4 shows the microcontroller mounted on the controller board.

3.2.4 Bluetooth Module

I use the AMB2300 from Amber Wireless[4] as Bluetooth module. The module is an integrated solution based on the National Semiconductors LMX9830 Bluetooth chip. It has an integrated antenna and offers a serial port interface for transparent data transfer. It is a class 2 module and operates to a maximum range of 20m. The theoretical maximum data rate over the air is 704 kbps. Since we are dealing with a radio transmission, the data rate is not guaranteed in practice. But the experiments show, that it works fine up to a few meters in different surroundings. Figure 3.4 shows the module mounted on the controller board.

[4]Handbuch AMB2300, BlueNiceCom4, Version 2.2, www.amber-wireless.de

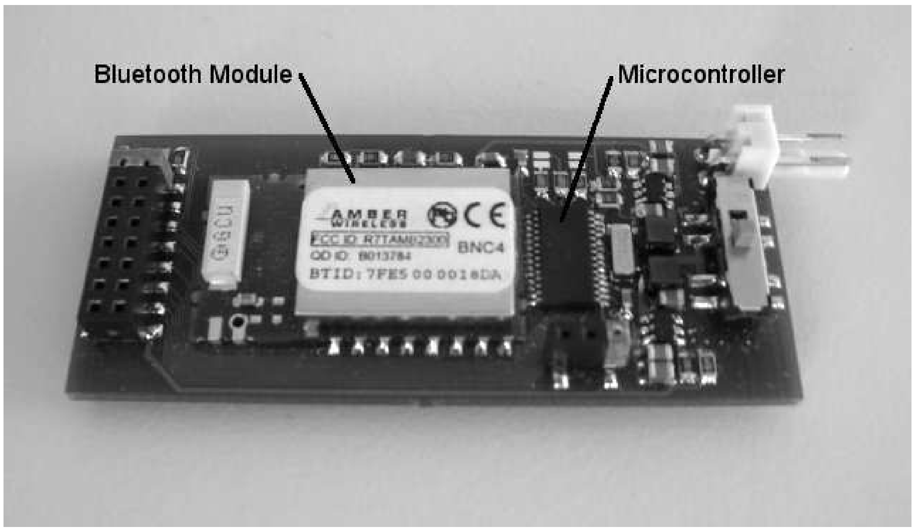


Figure 3.4: The controller board.

When used as slave device, the module has two internal states, transparent mode and command mode. It starts up in command mode. In this mode, commands can be sent to the Bluetooth module over the serial interface. No data is sent over the air. When a Bluetooth master device connects to the module and a link is established, transparent mode is automatically entered. In this mode, all data that is sent to the module is transferred over the Bluetooth link byte by byte .

3.3 Implementation

In this section, I give a more detailed view on how the glove device was actually implemented. I describe the basic circuit layout and the microcontroller program from a functional point of view.

3.3.1 Circuit Layout

The circuit design and layout was done by Michael Mende from the Institute for Process Control and Robotics (IPR), University of Karlsruhe. Physically, the device consists of three parts, the sensor board, the controller board and the battery pack. This split in seperate parts offers the greatest flexibility and the sensor board, which has to be mounted on the back of the hand, can be made as small as possible. Consequently, the sensor board only holds the sensor and a connector, through which it can be connected to the controller board. This is done by an 8-pole flat cable. The controller board holds the microcontroller, the Bluetooth module and all other electronical components besides the power supply. As power supply, I use a battery pack including two AAA batteries.

Sensor Board

The sensor board is shown in figure 3.3. It contains the sensor itself, the connector for the controller board, a red LED[5] and a 3-pole connector for an external analog source. The sensor has an internal AD converter, so for example, a resistive wire

[5]Light Emitting Diode

strain could be attached to the sensor. For my experiments, I did not use the additional AD converter. The LED is connected to an IO port of the sensor, which can be programed, so it can be set or unset by the microcontroller program. The sensor board circuit diagram is shown in figure A.1 and the circuit layout in figure A.2.

Controller Board

The controller board is shown in figure 3.4, it contains all other components, except the power supply. The circuit diagram can be found in figure A.3 and the circuit layout in figure A.4. Besides the microcontroller and the Bluetooth module, the controller board contains a 16-pole connector, which offers the connection of two sensor nodes. Only one was used in this work and the microcontroller program can deal with only one in its current version. There is a yellow and a green LED next to the connector, which can be used to indicate, if sensor boards are properly connected and functioning. There is a red LED, which is connected to the Bluetooth module and is controled by it. It indicates, if data is sent or received over Bluetooth. There is a 3-pole connector interface for programming and debugging the microcontroller. It is a proprietary interface called Spy Bi-Wire developed by TI. To program the microcontroller, the eZ430-F2013 Programmer from TI was used. This programmer comes in form of a USB stick and can be connected by a cable with the connector on the controller board. Development as well as debugging is possible over this interface. The integrated development environment Code Composer Essentials[6] was used to program the microcontroller. This is an Eclipse based commercial product from Texas Instruments, which is available in a free version, limited to progams of 16K maximum size. There are two voltage converters based on the Texas Instruments TPS61070 integrated circuit. Both are step-up converters, i.e. the input voltage can only be raised and not lowered. The converters produce 5 V and 3.3 V supply voltage for the circuit. This limits the use of alternative power supplies. The use of lithium-ion batteries with 3.7 V is not possible, since the voltage can't be lowered to 3.3 V. There is a power switch and a 2-pole connector for the power supply on board. Right after switching power on, the microcontroller will start executing its code. There is a voltage divider circuit, which has the unconverted battery voltage as input and outputs this voltage divided by two. This output is measured by the microcontroller through one of its IO ports, which can be used as analog-digital converter. The battery voltage gives information about the charge state of the battery. One should be aware of the fact, that voltage and charge state are not linearly correlated. One has to refer to the datasheets of the battery manufactorer or just measure it oneself. For the used tecxus[7] NiMH rechargable micro (AAA) batteries with a capacity of 1100mAh, the glove runs for five hours in send mode lying next to the receiving laptop.

3.3.2 Software Architecture

On the software side, the microcontroller program consists of an infinite main loop. Figure 3.5 shows the structure of the program. After some initialization commands the microcrontroller enters an infinite loop, which is executed until the power is

[6]Code Composer Essentials (CCE) Core-Edition v3.1, Texas Instruments, www.ti.com
[7]www.tecxus.com

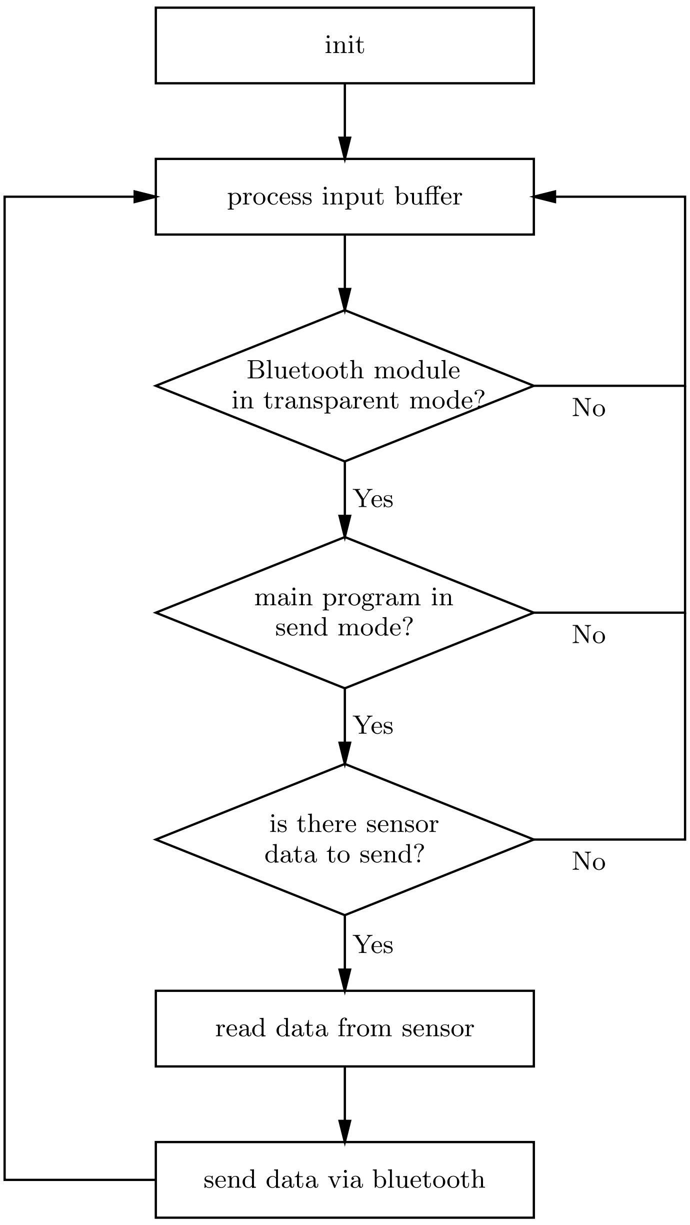

init
process input buffer
Bluetooth module in transparent mode?
No
Yes
main program in send mode?
No
Yes
is there sensor data to send?
No
Yes
read data from sensor
send data via bluetooth

Figure 3.5: Main program control flow chart.

switched off. The program has two main states, it can be either in command mode or in send mode. In command mode, application commands can be send to the device to request information or to alter device settings. In this mode the device does only send data on request. In send mode, the device continuously sends sensor data, while ignoring any commands, except the one for leaving send mode.

In send mode, every time, new data from the sensor is available, it is read and directly sent over the Bluetooth link. When the sensor has a new data sample, it pulses the first of his IO ports, which triggers the execution of an interrupt service routine on the microcontroller. So every time, new data is ready, this routine will be executed. It increments an internal sample counter and sets a variable to indicate, that new data is available. In the main loop, this variable is continuously checked and if set, the data is read from the sensor and send to the Bluetooth module. The device has two different data processing modes. In raw mode, data is passed to

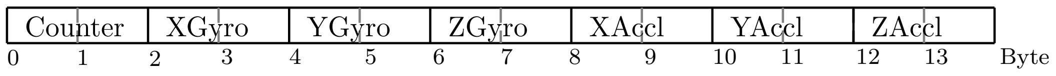


Figure 3.6: Structure of a data packet. This is the format of the packets sent over the Bluetooth link. Every data packet is 14 byte long

the Bluetooth module and thereby to the receiving computer, unchanged as read from the sensor. In intelligent data mode, the data is converted to a more friendly format before sending. To understand the need of such behaviour, one has to look at the format of the data, the sensor delivers. Every read operation returns a 16 bit data value. But the actual sensor data comes in 14 bit two's complement format. This is encoded in the 14 least significant bits of the 16 bit data word. Whereas the most significant bit indicates the availability of new sensor data and the second most significant bit indicates, if an error occured or not. So in raw data mode, the application processing the data, has to take care of extracting the 14 bit value and convert it to a standard machine format. In intelligent data mode, the conversion is done on the microcontroller and the application doesn't have to deal with this. In the current version, the information from the two most significant bits is just skipped in this mode, that means no information on the error state or the availability of new data is transferred. Either way, the sample counter and afterwards the six sensor data values are directly sent to the Bluetooth module. The data is sent in little endian byte order. Figure 3.6 illustrates the format of one data packet, every packet is 14 byte long.

In command mode, the device ignores new data from the sensor. It only acts on data, that comes from the Bluetooth module. In this mode self defined application commands can be sent to the device over the Bluetooth link. The device executes the command and sends back the result. The format of the commands is the same as the message format of the Bluetooth module. The Bluetooth module itself sends status messages to the microcontroller about its internal state. Since these messages have to be decoded anyway, it is convenient to use the same format for the application level commands as well. So the packet processing code can be shared. All data from the module is saved in a buffer and processed by the main program. In figure 3.5 this is represented by the block *process input buffer*. The incoming data is saved in the buffer until a complete message is received. Then, it is decoded and an appropriate reaction is performed by the program. As written in section 3.2.4, the Bluetooth module also has two internal states, which of one is also called command mode and one is called transparent mode. One has to be careful not to confuse these modes. The Bluetooth module listens for commands, when in command mode and works as serial interface, when in transparent mode. The glove device main program also listens for commands from the controlling computer, when in command mode and sends sensor data, when in send mode.

3.3.3 Communication Protocols

The communication with the sensor is straightforward. The sensor acts as SPI slave device, i.e. the communication is initiated by the microcontroller. This is done by setting the chip select (CS) line to active low and applying a clock signal on the SCLK line. Every cycle, 16 bits are transmitted. Data is read from the sensor by

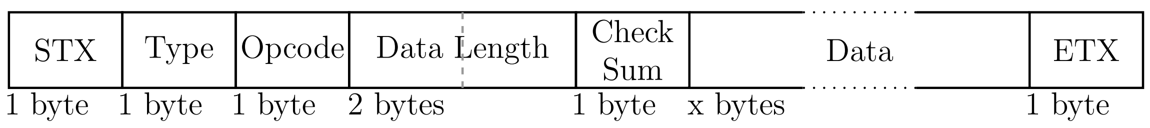


Figure 3.7: Structure of a Bluetooth message, also used for application commands. The STX and ETX bytes mark beginning and end. The length of the data field is given by the field "Data Length".

sending the appropriate register address in one cycle and read the corresponding data in the next cycle. The interface operates in full duplex mode, which means that the next register address can be sent, while reading data from the last one.

Communicating with the Bluetooth module is more complicated, since messages do not have constant length and data is not only sent on request. This communication link is also used by the computer application to send commands to the device. These self defined commands have the same structure as the Bluetooth module messages allowing easier processing. Figure 3.7 shows the format of the messages. The Bluetooth module supports a lot of different profiles and modes, there is a complex communication protocol implemented. But since I am only using the serial port profile (SPP) in slave mode, there are only a few messages of interest.

When data is sent from the Bluetooth module to the microcontroller, an interrupt service routine is executed which reads the data and writes it to a circular buffer. The buffer works as FIFO with limited preconfigured size. In the main loop of the program, this buffer is continuously checked for data (see figure 3.5: process input buffer). It's assumed that all data read from the Bluetooth module has the packet format shown in figure 3.7, otherwise the program runs into an error state and must be restarted. The data is read from the buffer byte by byte and the packets are reconstructed and processed. Four different messages from the Bluetooth module are relevant for the application. These are shown in table 3.1. I defined some application commands, that are all of type `APP`, i.e. they use a so far unused number in the type field of the packet (see figure 3.7). These commands can be used by the application on the computer side. They just have to be written byte by byte to the serial port, on which the Bluetooth link is acting. Table 3.2 lists these commands. When in send mode, all commands except `APP_STOP_SENDING` are ignored. The device starts up in command mode.

`LMX9838_READY`	Indicates that the Bluetooth module is operational and ready
`SPP_INCOMING_LINK_ESTABLISHED`	Indicates that a Bluetooth master has established a link with the module. The module is now in transparent mode.
`SPP_TRANSPARENT_MODE`	Indicates that transparent mode was left
`SPP_LINK_RELEASED`	Indicates that the link to the master is down

Table 3.1: Bluetooth module messages of interest.

Message Name	Description
`APP_START_SENDING`	Switch device to send mode
`APP_STOP_SENDING`	Switch device to command mode
`APP_ADIS_READ`	Read register adress from sensor. 8 bit address is provided in data field, the 16 bit value of the address is back.
`APP_ADIS_WRITE`	Write 8 bit value to given address. The data field of the packet consists of 2 bytes, the 8 bit address in the first byte and the 8 bit value in the second byte.
`APP_GET_BATTERY_VOLTAGE`	Battery voltage is sampled and sent as 16 bit little endian value over Bluetooth. The unit of the value is Volt.
`APP_ECHO`	Echo service for testing purpose, sends back all data fields in the same order.
`APP_GET_ID`	An id string with version information is sent back. At the time of this work, this is “InertialBlue V0.1”.
`APP_SET_DATA_MODE`	Sets the data mode by sending 0x00 (raw data mode) or 0x01 (intelligent data mode) in the data field.

Table 3.2: Self defined application device commands

3.3.4 Device Driver

On the receiver side, communication with the device is done through a virtual COM port. If the device is switched on and the application opens the COM port, the Bluetooth link is automatically established by the operating system. The application can read and write to the serial port just like every other serial port. I implemented a python class, which encapsulates the basic functionality of the device. This implementation is not complete in a sense, that every possible function is implemented, it only provides the demands of this special application. So, for future applications of the device, enhancements of the implementation may be necessary. The class offers methods to start and stop the device, sending commands and reading data from it and to calculate the number of missed packets. On the base of this class, I implemented a threaded module for integration into the BiosignalsStudio, a software developed at the Cognitive Systems Lab. The framework offers a unified data stream format and modules to save received data to files or to visualize the received data. All recordings of data were done with this software.

4. Handwriting Recognition

In this chapter, I analyze the given problem of handwriting recognition in the air in more depth. The differences to the field of conventional handwriting recognition are examined. I also take a closer look on the signals, namely accelerometer and gyroscope data, which we are dealing with here. Reasonable preprocessing steps are infered from the analysis of the signals.

4.1 Airwriting vs. Handwriting

Conventional handwriting recognition deals with a lot of different tasks. For example there is the field of signature verification and writer identification. Here, the focus is on recognizing writer specific properties to discriminate who has produced or is producing the writing. The biggest research area is for sure the task of finding the actual transcription of a handwritten text. This field itself is divided into off-line and on-line handwriting recognition. In the off-line case, an already written text is the subject of investigation and only the graphical result of the writing process is available. Typical areas of application are the recognition of scanned hand written documents or forms. On the other side, there is the task of on-line recognition, where text is recognized during the process of writing. In this case, additional information is available, mainly the trajectory, including speed, pen-up and pen-down information. This means, one knows, if the writer is actually writing on the surface or if he or she is just moving the pen to a new position. A typical input method for on-line hand writing recognition is a tablet PC. As mentioned in section 2.3, there has been extensive research in these areas.

The recognition of handwriting in the air is different, but in principal we are dealing with an on-line recognition problem, since we get signals while writing. On the signal side, there are two major differences to a conventional system. We do not get the trajectory directly and we do not have the pen-up and pen-down information. The missing pen-up and down movements result in a single continuous stroke for the whole writing. For sure, there are pauses in writing, but these have to be extracted from the data and do not correspond with the missing pen-up and pen-down information. In principal, we get a continuously unsegmented data stream

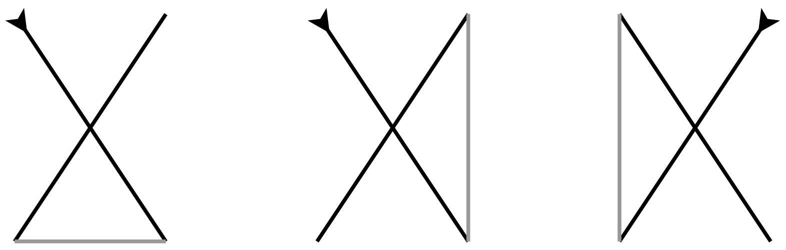


Figure 4.1: Writing variants of the character X, grey strokes mark motions, which would be pen-up movements in traditional writing. All variants were observed on real test persons.

from the sensors. Even several sentences would yield one single stroke. This makes the task more difficult, since pen-up and down movements automatically give a segmentation of the data. In the case of continuous handwriting recognition, this is not naturally a segmentation of characters, but surely gives useful information, we are missing.

Beside the segmentation problem, the missing pen-up and down information leads to another problem. In conventional handwriting, as well as in our case, differing individual writing styles are a problem. The trajectories of characters differ when writing with different stroke sequences. As example one can think of the character "X". Exactly the same X can be written by starting in the upper left corner or in the upper right corner. This obviously leads to different trajectories and thereby to different acceleration and angular rate signals. Figure 4.1 shows three different writing variants of the character X, leading to different signal sequences. In the case of *airwriting*, not only the order of the strokes differs, but also the actual shape of the characater, shown by grey strokes in the figure.

Furthermore, the problem with raw inertial sensor signals is, that they are also highly dependent on how a character was created. Not only the temporal sequence of strokes, but also the smoothness of writing is encoded in these signals. When a character is written in the same sequence of strokes and the shape is nearly identical, there still could be differences in the signals from the inertial sensor. One person might write very smooth, while the other one writes very choppy. I will discuss this issue in more detail in section 4.2. The reconstruction of the trajectory might solve this issue, since recognition based on trajectories does not have to deal with the speed of writing. But on the other hand, there might also be advantages of using raw inertial sensor signals over trajectories. Since the raw signals contain more information about the individual writing style, a writer adaptive system might reach better recognition performance. A writer adaptive system is a writer-independent system, which adapts itself automatically to the individual writer. In [OCB^{+}04] Oh et al. use a pen style device for airwriting and implement a trajectory reconstruction. They reach better results for the inertial signals opposed to the estimated trajectory on single digit recognition. But they also note, that the inaccuracy of the trajectory estimation caused by the sensor drift, might account to the lack of performance. In [KCK06] Kim et al. propose a method to model the pen-up states. They segment the given 3D trajectory in strokes and identify pen-up strokes with Bayesian Networks. The pen-up strokes can then be removed from the trajectory and as result traditional handwriting recognition techniques can be applied, and existing training data can

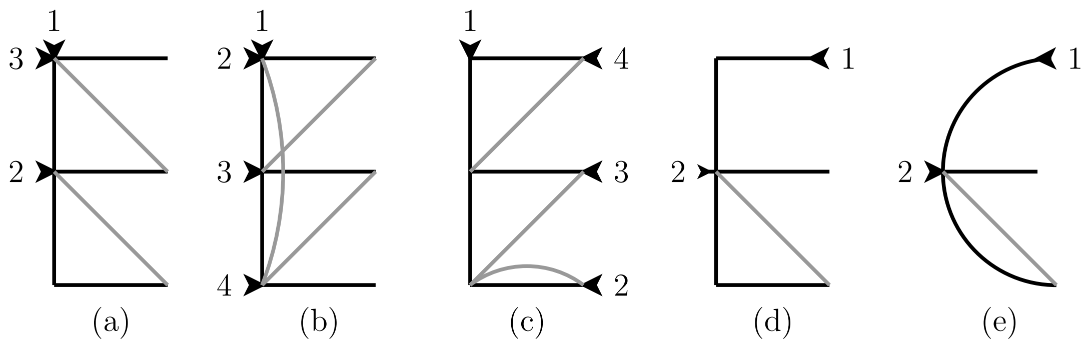


Figure 4.2: Different variants of writing an E seen among ten test persons. The grey strokes correspond to movements, which would be performed with pen-up, when writing on paper. Variant (a) was observerd on 4 test persons, variant (b) on 3 and variant each of (c),(d) and (e) on one.

be used. While their results are good, it is hard to compare them to my results, since they give little information on details of their experiments.

Another aspect is the absence of visual feedback. When there is no trajectory reconstruction performed, there is no possibility to give the writer visual feedback, on what he or she is writing. Even when the trajectory is available, we might not have the possibility, or do not want to give visual feedback. This is a realistic scenario when it comes to outdoor use for example. Without visual feedback, we can expect the shape of the characters to be less accurate than with visual feedback. One can see this effect, when writing on a paper with closed eyes. Especially with characters, that have the same basic movement and only differ in starting or endpoints of strokes, this could lead to recognition difficulties. The digits 0 and 6 are good examples for such characters. In principal, they only differ in the height of the stroke endpoint, relative to the starting point.

4.2 Individual Differences in Writing

As I already noted in the section above, different sequences of strokes in characters are a problem. The test persons were observed while writing in the air to get an impression, which characters have varieties in the sequence of strokes. Figure 4.2 shows different writing variants of the character E. The grey lines show the movements between the strokes, in traditional handwriting, this would be pen-up movements. To face this problem, a higher number of Gaussians in the mixture model might work. This assumption is supported by the results of the experiments on character recognition in section 5.4.3. Thereby, one HMM can model different writing variants. Another possibility would be to train individual models for the writing variants. The observation of the test persons suggests, that most characters are generally written in the same stroke sequence. So, training different models would be affordable. Though this approach was not implemented in this work.

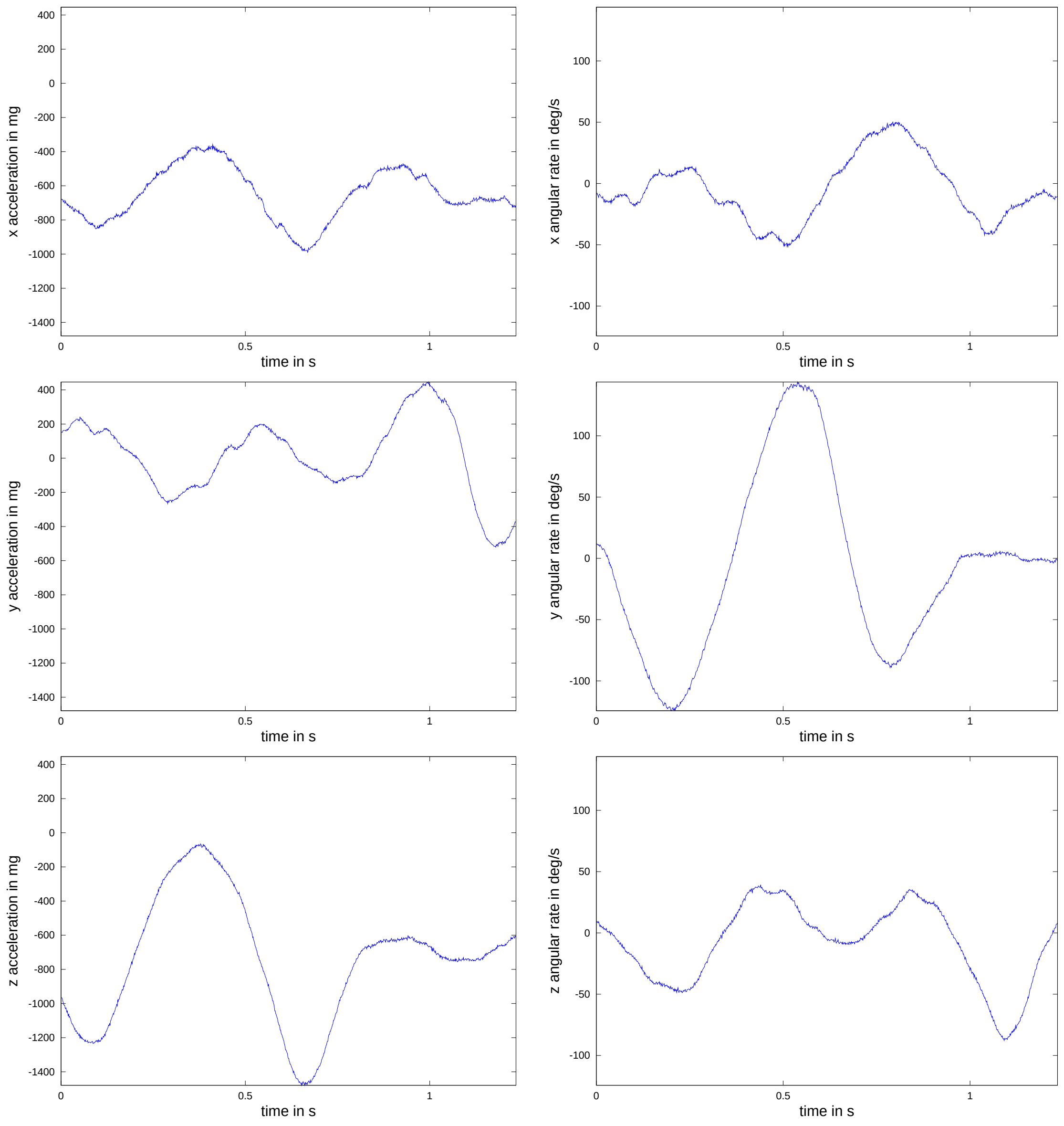


Figure 4.3: Inertial sensor signals for the character A. The left column shows accelerometer data, the right column gyroscope data

4.3 Signal Analysis

We will now take a closer look on the actual signals that the sensor delivers. Thus, we are able to infer reasonable preprocessing steps. Figure 4.3 shows all sensor channels for an example writing sequence of the character “A”. At first glance we can see that signals are already very smooth, so sensor noise is low compared to the dynamic range of the signals. Application of extra smoothing filters might not be necessary. Figure 4.4 shows the orientation of the sensor coordinate system relative to the hand used for writing. We can get an impression which axis acounts to which kind of movements. But direct interpretation of the signals in sense of trajectory estimation is hard. We can not easily look at the signals and understand how the curves result in the specific letter A. We should be aware that the coordinate axis of the sensor are not aligned with the coordinate system of the imaginary blackboard, we are writing on. Besides, we have parts of gravitational acceleration in all three accelerometer signals. And as one can see from the gyroscope signals, the attitude

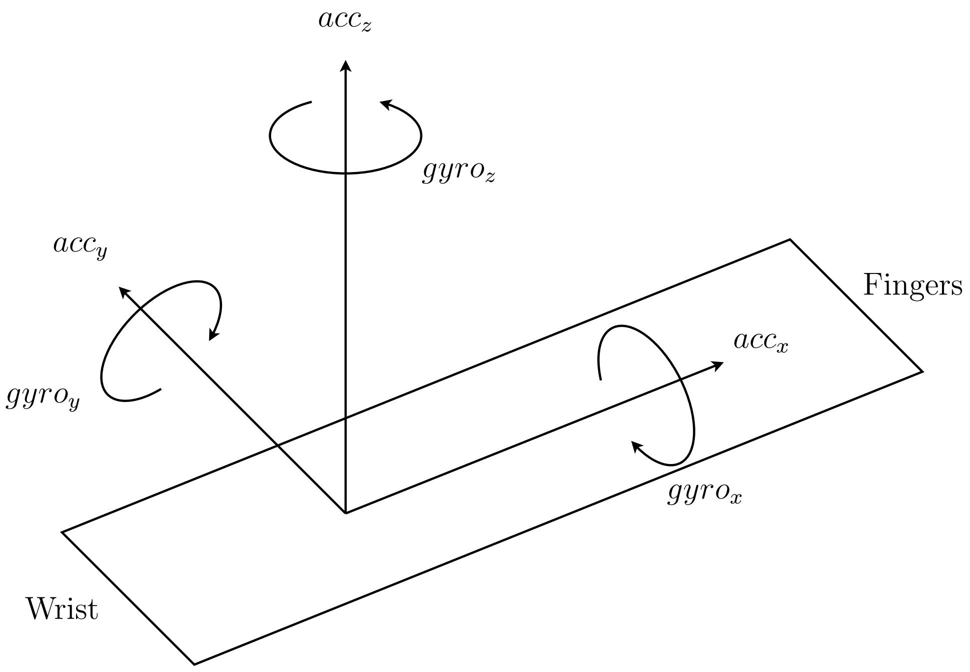


Figure 4.4: Orientation of the sensor coordinate system on the hand.

of the sensor is not constant, but does change over time, and thereby does the distribution of the gravitational acceleration over the three axis. In Figure 4.3, we can see, that the dynamic range of acceleration in x direction is less than that of the y and z directions. This makes sense, since the largest part of measured x acceleration would be caused by a movement perpendicular to the imaginary writing plane. We would expect little movement in this direction.

We can also tell something about the segmentation quality from the signals. In section 5.1 and 5.2 I will explain the experiment setup in detail but for now, one should know that the test persons segment their own writing by pressing keys on a keyboard at the endpoints of the motion. Ideally, while pressing the key, they are not moving at all. If the sensor is not moving, only the gravitational force is measured and thus, the following equation must hold:

$$1000\ mg = \sqrt{(acc_x)^2 + (acc_y)^2 + (acc_z)^2}\ mg \tag{4.1}$$

If the equation holds, this does not prove, that the sensor is not moving, but if the equation does *not* hold, the sensor *is* moving. When looking at the accelerometer signals in figure 4.3, we can see that first, equation 4.1 does not hold and second, the gyroscope signal are not zero in the beginning. So, the test person was already moving, when pressing the start key and thus segmentation quality is not perfectly accurate in this case. This is also the case for other investigated character signal sequences and conincides with the observation of the test persons. In general, we can not rely on an accurate segmentation, which might have implications, when building word models from the character models, but this is discussed in section 5.5.

4.4 Preprocessing

We have analyzed the sensor signals in the last section and are now able to infer a reasonable preprocessing. First of all, there is the already noted problem of gravitational force (see also section 2.1). The gravitational force can be assumed orthogonal to the earth surface, thus is constant in direction and magnitude. So, depending on the sensors attitude relative to the earth surface, the sensor always measures the x,y and z components of the gravitational acceleration. The attitude of the sensor definitely is not the same over different persons and sessions and also while writing. As we can and want not force the user to permanently maintain a constant attitude with his hand, we need to remove or compensate the effect of the gravitational force. One possibility would be to track orientation of the sensor with the gyroscopes like in an inertial navigation system (see section 2.1.2). If orientation is known, the gravitational acceleration can be removed from the signal. This approach has two drawbacks. First, gyroscope data must be integrated and thus, small sensor errors accumulate over time, resulting in an errornous attitude calculation. Second, the initial attitude must be known, which would force the user to assume a defined and accurate starting position with his hand, which is not desirable. The approach I took relies on the assumption, that the hand attitude does not change much during writing. If this is true, we can think of the gravitational acceleration as a constant offset in the signals. By subtracting the mean of the signal sequence, this offset is removed. The variance will also be normalized to reduce the effect of different writing speeds.

Filtering is another aspect of preprocessing. Our signals are clearly time domain signals. A transformation to frequency space does not seem reasonable. For time domain signals, a moving average filter is a possible choice. It normally gives better results, than a low pass filter, which is optimized to have good properties in the frequency domain. As we have seen, the signals are already pretty smooth, so one should not expect too much from filtering. In fact, experiments showed, that filtering the signals does not have a significant positive effect. Mean and variance normalization and smoothing are surely not the only reasonable preprocessing steps to try. Extracting maxima and minima as features or calculating the delta coefficients might increase the performance of the recognition system. Since the focus of this work is to evaluate the possibility to build a live word recognizer, no further effort was made to increase recognition performance by trying different preprocessing steps.

4.5 Recognition

In section 2.2.2, I already gave a brief outline on how an HMM based recognizer works. In this section I give a more detailed description of such a system. First of all, we will look at our observation sequence. The observation sequence consists of the preprocessed signals, in our case, every sample corresponds to one observation. By sample, I mean the values of the accelerometer and gyroscope sensors at a specific point in time. Thus, every recording of a digit, character or word consists of a finite amount of samples, depending on the sampling frequency. Let's assume we have a reasonable preprocessing for our system and thus do have a preprocessed signal sequence. We need to define the HMM topology first, which in our case means to define the number of states of the HMM since we use left-right models. This topology will be unchanged for the rest of the process. We again denote the number of states we use by N. The next task would be to initialize the HMM parameters with some meaningful parameters to speed up and improve the quality of the training process. We have to find initial parameters for the Gaussian mixture models for

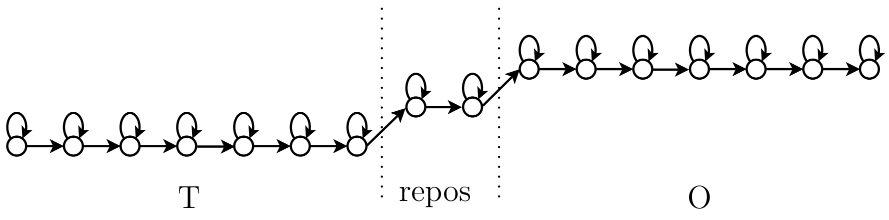


Figure 4.5: An example concatenated model for the word “to”. It consists of the independent models for the graphemes “T” and “O” with 7 states and the 2 state model for the repositioning in between.

every state. Basically, it is possible to use random parameters here but by using the recorded data, we can do better. The problem is, we do not yet know, which samples correspond to which state of the hmm model and by that do not know which samples to take into account for calculating initial parameters for the gaussians for every state. But since we have a left-right topology with no possibility to jump back in the HMM model, the states are temporally ordered and thus we can simply map the observation sequence to the sequence of states by splitting up the preprocessed sample sequences into N parts of equal length and map every part to one state. For sure, not every state will correspond to exact the same amount of observation sequences, but it is a reasonable starting point. This strategy is successfully applied in speech recognition and is refered to as “flat start”.

But yet, we only have a mapping between samples and states and no initial parameters. Like the HMM topology, the number of Gaussians per state is defined in advance. Let’s denote the number of Gaussians per state by k, i.e. we have a GMM with k components per state. For every mixture component, we need to initialize the mean vector and the covariance matrix, which is assumed to be a diagonal matrix. This can be done by clustering the samples in as much clusters, as we have Gaussians in the mixture models. To be more precise, to initialize parameters for one specific state, we take all samples from all training signal sequences associated with this state. All Samples are then clustered using the k-means algorithm. As result, we have one cluster of sample values for every Gaussian. We simply take the mean and variance from this cluster to initialize the Gaussian. We do now have a fully initialized system, based on which the training procedure can be started. This is done in several training iterations. In every iteration, all sample sequences in the training data set are processed. Every signal sequence is preprocessed and afterwards used to train the recognizer. On base of the viterbi algorithm, the Baum-Welch algorithm is used to reestimate the parameters. The path with the highest probability of producing the observation sequence is computed using the viterbi algorithm. The parameters of the Gaussian Mixture Models are reestimated to maximize the likelihood of this observation sequence using the Baum-Welch algorithm.

4.5.1 Word Recognition

Assuming we now have a trained model for each character, we can connect them to build word models and thereby build a word recognizer. Our recognizer would have a pre defined vocabulary, containing all words that could be recognized. For each word in the vocabulary the HMMs of the individual characters of this word are linearly connected to a new HMM. A recognizer with a vocabulary of, for example, 100 words would have 100 word models, each of which is a concatenation of individual character models. One thing, such a HMM does not model very well, are the motions between two characters. There is always some kind of repositioning between the individual characters. These reposition movements should also be modeled and the corresponding HMMs are inserted between the character HMMs. Figure 4.5 shows an example model for the word *TO*. When an unknown signal sequence is presented to the recognizer, the probability of every word model, given the signal sequence is calculated and the word with the highest one is output as the hypothesis.

5. Experiments

I performed a series of experiments to evaluate the data glove in conjunction with the recognizer. Experiments were conducted on digit recognition, character recognition and word recognition. For the task of character recognition, data was recorded from different test persons. For the first experiments in digit recognition, analog wire bound sensors were used, because the data glove was not yet available. In this chapter, I give an overview of the experimental set-up, as well as a detailed description of all performed experiments. The results are presented and analyzed, and possible solutions for arising problems are given.

5.1 Software Set-Up

Generally the software setup consists of a software for recording the sensor signals and a recognizer to train and test models on the recorded data. I used two different software systems for recording the raw acceleration and gyroscope signals, one specialized on analog signals and one specialized on the data glove. Both systems can record several signal channels simultaneously and write the recorded data to files. I used DiaMess[1] for the analog sensor recordings. The data glove signals were recorded using BiosignalsStudio, which is a signal processing, recording and visualization software framework, developed at the Cognitive Systems Lab. As mentioned in section 3.3.4, I implemented and integrated a device driver for the data glove into BiosignalsStudio. Within the BiosignalsStudio framework I wrote different recording front ends. A text based front end, as well as a graphical user interface. In all versions, the segmentation of the recording must be done manually by the user. The software prompts for each digit, character or word to write and the recording can be started and stopped with a key press. This method of segmentation is assumingly not as accurate as, for example, a post record segmentation with the help of a parallel recorded video stream. But the effort of segmentation and recording is much lower, which makes data acquisition a lot easier. The HMM modeling is implemented using the Janus Recognition Toolkit (see [FGH+] and [SMFW01]). Janus was originally designed for speech recognition but can be used as a general purpose HMM recognizer.

[1]DiaMess was developed at the Institute for Sports and Sport Science, University Karlsruhe

5.2 Experimental Set-Up

The set-up of the experiments was basically always the same, only in case of the wire bound sensors, a slightly different set-up was used. I describe the differences in the set-up of these early experiments in section 5.3.1 in conjunction with the results. For all experiments performed with the data glove, the same set-up was used. One recording session is generally denoted by the term session. During a session, the glove ist not taken off and on, therefore the sensor position on the back of the hand can be assumed to be unchanged. Also the general setting is not changed during one session, i.e. the same chair and table is used, the laptops running the recording software is not changed in position and only short pauses (approximately 5 min maximum) are made. Different sessions are denoted by $X_{[d,c,w],n}$, where X denotes the Writer, d, c or w indicate the type of recording, i.e. digit, character or word, and n denotes the session number, i.e. $A_{c,2}$ means character recording session 2 by writer A.

5.2.1 Recognition Tasks

Experiments were conducted on digit, character, and word recognition. Different settings to investigate session dependency and writer dependency were evaluated, for which I introduce the following Terms:

- Session-Dependent (SD): The training, development and test data is taken from the same recording session. This implies writer dependency, since one session can only have one contributing writer.
- Multi-Session (MS): Data from two or more sessions is merged and training and test data is taken from this merged data set. Such a system incorporates data from different recording sessions, but is not session independent, since the recognizer uses data from all sessions for training. This implies writer-dependency.
- Session-Independent (SI): Training and test data is taken from different recording sessions. This implies writer-dependency.
- Writer-Dependent (WD): Training and test data originates from the same person.
- Writer-Independent (WI): Training and test data originates from different writers. The training as well as the test data can consist of more than one writers data.

For the session-dependent, multi-session and writer-dependent setting, the data was divided into a train set, a development set and a test set. The recognizer was trained on the train set and evaluated on the development set. The best performing recognizer on the development set was then again evaluated on the test set. The evaluation on the test set gives the final result. For the session-independent tests, the optimal number of HMM states and Gaussians per state from the session-dependent test were used. Therefore no additional development set was necessary. For the writer-independent setting a *leave-one-out cross-validation* (LOOCV) was used. This means taking the data from one writer as test set and the data from the other writers as training data. This is done for every writer, such that the data from every writer is once used as test set. The final evaluation result is the average of the individual results with the same parameter setting.

The influence of the different sensor channels on the recognition performance was also investigated. In general, the used sensor channels are given by $a_{[xyz]}$ for the accelerometer and by $g_{[xyz]}$ for the gyroscopes. For example, a_{xy}, g_z indicates the usage of the x and y accelerometer and the z gyroscope channel.

The performance of all systems is given by the recognition rate, which is the percentage of correctly classified reference recordings in the train set. It is defined as

$$\text{recognition rate} = 100 \cdot \frac{\text{number of correctly classified references in test set}}{\text{total number of references in test set}}. \quad (5.1)$$

A reference is one recording, i.e. one digit, one character or one word. For example if the test set contains 650 characters, of which 10 are misclassified, this equals a recognition rate of $640/650 \cdot 100 \approx 98.46$. The results are always rounded off to two decimal places. It should be noticed, that in case of word recognition, one reference equates one word, the recognition rate gives the percentage of correctly classified words and not the percentage of correctly classified individual characters.

5.2.2 Writing Style and Segmentation

The test persons were always sitting on a chair and were asked to write in front of them like writing on an imaginary blackboard. The writing hand was approximately at eye level. All writing was supposed to be done keeping the wrist fixed, which is quite natural. Writers were asked to write characters between 10 to 20 cm in height. Furthermore the subjects were told to write in place, which means, the horizontal position should be approximately the same for every character written.

In handwriting recognition one discriminates between cursive and block writing. All writing in my experiments was done in block letters and furthermore only capital letters were allowed. The test persons did not have to write perfectly formed letters, they were just told to write in the same manner, as they would write on a blackboard in capital block letters. That means, the test persons were free in the choice of how they write the characters under the constraint of capital block letters, but they were asked to be as consistent as possible in the way they write the letters. This also applies to the word recognition task, i.e. the words were written in capital block letters. There were no experiments on cursive handwriting recognition.

As explained in the previous section, the segmentation of the recordings was done manually by the test person while recording. The proband had to press a key, before starting and after finishing a digit, character or word. This key press was performed with the other hand than the one used for writing. All test persons were told to have their writing hand in start position for the first stroke of the individual digit, character or word and to hold it still, when pressing the start key. The second keystroke marks the end of the writing motion. Again the test persons were told to hold their hand still and press the key right at the end point of the motion, before performing any movements not belonging to the digit, character or word, including moving their hand to the next start position.

5.2.3 Modeling

The digits and characters are always modeled by one individual HMM model. In the case of word recognition, the character models are concatenated to form a word model as described in section 4.5.1. To represent the repositioning between the individual characters, a repositioning HMM is inserted between the individual character models. The number of states for a character or digit model is the same for all models in every individual experiment. But for different experiments, different numbers of states are used to investigate the influence of the number of states on the recognition performance. The same applies to the number of Gaussians, which is the same for every state and over all HMMs. Again, for different experiments, this number is varied too. For example, there might be two experiments, one, in which every character model is composed of eight states and every state has a GMM with three Gaussians and another experiment, in wich every character model is composed of twelve states and every state has only one Gaussian. There were no experiments on settings like, 50% of the characters have models with eight states and 50% have models with twelve states.

In the description of the parameter settings, I also give the total amount of Gaussians per model, which is calculated by multiplying the number of Gaussians per state by the number of states. This gives an impression, how many parameters have to be trained.

5.3 Digit Recognition

The first row of experiments was conducted on single digit recognition. Since the data glove was not yet available, this was done using wire bound 3-axis accelerometers. After the data glove was finished, the experiments were repeated with this device.

5.3.1 Analog Sensor Approach

The very first experiments with the analog sensors were done to get an impression of the difficulties of the given task of handwriting in the air and to have realistic data available, when implementing the recognizer with Janus. I used Bosch 3-axis acceleration sensors, no gyroscopes were used. The sensors were wirebound and an external AD converter connected to the computer had to be used, in order to record the sensor signals. I attached one sensor to the back of the hand and another one to the forearm using tape. The writer segmented the recording manually by pressing a pushbutton switch connected to the AD converter. This trigger channel was later used to cut the recorded signal into pieces, containing the recordings of the individual digits. Before starting the first experiment, every digit was written on a sheet of paper (DIN A4) and the paper was attached to a wall in the height of the shoulder of writer A. The writer wrote the digit in the air by following the template stroke on the paper with his hand. This should ensure very similar movements for every digit. In the second experiment, this restriction was removed. Table 5.1 shows the results of the session-dependent evaluation of the first two experiments. No evaluation of a session-independent system was performed since there were differences in the experimental setup. The recognition rates are in the same range, as the latter results on digit recognition with the data glove. One can see, that taking only the data from

Sensors	Data Train/Dev/Test	States	Gaussians State	Gaussians Total	Rec.Rate in %
hand	$A_{d,1}$(200/150/150)	10	1	10	100.0
arm	$A_{d,1}$(200/150/150)	15	1	15	99.3
hand+arm	$A_{d,1}$(200/150/150)	8	2	16	99.3
hand	$A_{d,2}$(200/150/150)	10	3	30	100.0
arm	$A_{d,2}$(200/150/150)	12	1	12	100.0
hand+arm	$A_{d,2}$(200/150/150)	10	2	20	100.0

Table 5.1: Recognition results of digit recognition on the data of two sessions of the same writer. The table gives the results and the best performing parameter set for the different evaluations.

the sensor on the hand already leads to very high recognition rates. This motivated the construction of the data glove, since it seems sufficient to use one sensor on the back of the hand.

5.3.2 Sensor Glove Approach

The first experiments with the new data glove were done on digit recognition. Two sessions were recorded, both times with test person A. In the first session, 500 digits were recorded, in the second session, another set of 500 digits was recorded. In both sessions the same set-up was used, the data was recorded in blocks of ten digits, always containing the digits from 0 to 9 in their natural ordering. These blocks were repeated 50 times. On the recorded data many different recognizer configurations were tested. The recognizer was trained and tested on each individual channel to get an impression on how much every channel accounts to the performance of an overall system. A variety of parameters for the number of states in the HMMs and the number of gaussians in the mixture models were tested. The number of training iterations was varied and a moving average filter with different filter settings was used, i.e. the number of points, the moving average filter takes into account, was varied.

I examined recognizer performance for the session-dependent, multi-session and session-independent task. For the session-dependent and multi-session experiments, the data was randomly partitioned in three sets. The trainset contained 40% of the data, the development and testset 30% each. Each set includes the same amount of recordings for every digit. The best performing systems on the development set were chosen and evaluated on the testset. For the session-independent evaluation, the recognizer was trained on all data of one session and evaluated on the other session. The parameters, namely number of states and Gaussians, were chosen according to the optimum of the session-dependent evaluation. On the data of the first session, I performed tests on the recognition performance, when using only a single sensor channels. This means, only one of the channels $a_x, a_y, a_z, g_x, g_y, g_z$ was used. I also made experiments using only accelerometer or only gyroscope data.

Table 5.2 shows the results of the single channel, accelerometer only and gyroscope only experiments together with the results, when using all available channels (accelerometer + gyroscope). Notable is the already very high recognition ratio for

Task	Data	Sensor Channels	States	Gaussians State	Gaussians total	Training Iterations	Recognition Rate in %
SD	A_{1d}	a_{xyz}, g_{xyz}	8	2	16	20	100.0
SD	A_{1d}	a_{xyz}	10	1	10	10	100.0
SD	A_{1d}	g_{xyz}	12	1	12	10	99.3
SD	A_{1d}	a_x	12	2	24	10	77.3
SD	A_{1d}	a_y	12	4	48	10	87.3
SD	A_{1d}	a_z	12	1	12	10	99.3
SD	A_{1d}	g_x	12	1	12	20	76.7
SD	A_{1d}	g_y	6	3	18	10	92.0
SD	A_{1d}	g_z	12	4	48	30	88.7

Table 5.2: Recognition results of digit recognition with the data glove on data of the first recording session. The rows show the results of different sensor channel configurations. Different parameters for the number of states and Gaussians were probed for each of the experiments. The table gives the best performing parameter set on the development set for each experiment.

Task	Data Train	Data Dev	Data Test	States	Gaussians State	Gaussians Total	Rec.Rate in %
SD	$A_{1d}(200)$	$A_{1d}(150)$	$A_{1d}(150)$	8	2	16	100.0
SD	$A_{2d}(200)$	$A_{2d}(150)$	$A_{2d}(150)$	10	2	20	98.7
MS	$A_{1d+2d}(400)$	$A_{1d+2d}(300)$	$A_{d1+2d}(300)$	10	4	40	98.0
SI	$A_{1d}(500)$		$A_{2d}(500)$	8	2	16	94.2
SI	$A_{2d}(500)$		$A_{1d}(500)$	10	2	20	95.8

Table 5.3: Recognition results of digit recognition on the data of two sessions of the same writer. The table gives the results and the best performing parameter set for the session-dependent, multi-session and session-independent task. All recognizers were evaluated after 10 training iterations.

the z acceleration only setting. The used parameters were 4,6,8,10 and 12 for the number of HMM states and 1,2,3 and 4 for the number of Gaussians. All possible combinations were used to train a recognizer. The recognizer was evaluated after 10, 20 and 30 training iterations. A whole range of different settings resulted in the maximum recognition performance, the table lists the parameter combination with the minimum number of Gaussians in total out of all optimal combinations. I also made experiments, applying a moving average filter with 9, 29 or 59 points, which had no positive effect on the recognition performance. Table 5.3 shows the results of the session-dependent, multi-session and session-independent experiments. The same set of parameters as in the experiments on sensor channels was probed but each recognizer was evaluated after 10 training iterations and no moving average filter was applied.

All wrong hypothesis are caused by a confusion of 0 and 6. This is not surprising, since the motion performed when writing a 0 or 6 is almost identical. Writer A started 0 and 6 at the same point performing a circular motion afterwards. The writing only differs in the vertical end position of the motion. When writing a 0, one tries to end at the same position than one started, when writing a 6, one tries to end at approximately half of the character height. In [KCK06], Kim et al. also identified this ambiguity as one main reason for errors in digit recognition in their experiments. Since the test persons do not get visual feedback, they probably do not write as exact, as they would, writing on a real blackboard. Beside the existing similarity of the digits, this might also be a reason for this particular confusion. But without the real trajectory, it is not easy to figure out the exact reason, since we cannot really investigate, how the misclassified digits were really written. It is noticeable, that this problem did not arise for the experiments with the analog sensors, for which almost perfect recognition performance was reached. In this first experiments, the test person did write very accurate, in one case even with a printed template on the wall. In the experiments with the data glove, the test persons generally wrote more freely and thus less accurate. I assume that this is the reason for the problem of confusion between 0 and 6 in the data glove experiments.

5.4 Character Recognition

Character Recognition experiments were first performed for a writer-dependent setting on two recorded sessions from writer A. As a second step, data from ten test persons (A-J) was recorded, resulting in a dataset of ten different writers with 6500 character recordings in total. Nine of the test persons were right-handers, one was a left-hander. For each of the ten writers, a writer-dependent system was trained and evaluated. The results were used to investigate if the previously obtained results from witer A could be generalized to other writers for the writer-dependent setting. The data was also used to examine the performance of a writer-independet recognizer. For this purpose, a recognizer was trained on the data of nine subjects and tested on the data of the remaining subject (leave-one-out cross validation). Since the analysis of the results showed, that a system trained on right-hander data does not generalize very well on the data of the lefthand writer, a right-hander only system was also evaluated. In all experiments different settings for the number of HMM states and Gaussians were tried.

5.4.1 Data Recording

Character data from ten test persons, five female and five male, was recorded. One of the test persons is left-handed, the other nine are right-handed. The recording set-up was basically the same as in the previous experiments, but took place in different locations. The test persons were observed and the way they write characters was noted. The first four sessions were recorded using a software version, in which the alphabet is written iteration by iteration in natural order. In the last six sessions, a new software was used, which does not prompt for the ordered alphabet anymore, but for every character in the context of every other character. That means, every character is recorded 25 times and, for example, an “A” is once followed by a “B”, once by a “C” and so on until “Z”. This was done to exclude the possibility, that characters are written in context specific manner and that the test persons get used to the order. In the first four sessions, the alphabet was recorded 30 times, in the last six session 25 times. The data was reduced to 25 times the alphabet for all writers.

Test	Data Task/Train/Dev	States	Gaussians per State	Gaussians Total	Rec.Rate in %
SD	$A_{c,1}$(520/390/390)	8	2	16	99.5
SD	$A_{c,2}$(260/195/195)	12	2	24	98.5
MS	$A_{c,1+2}$(780/585/585)	12	5	60	97.1
SI	$A_{c,1}$(1300)/ - /$A_{c,2}$(650)	8	2	16	79.9
SI	$A_{c,2}$(650)/ - /$A_{c,1}$(1300)	12	2	24	88.9

Table 5.4: Results of the session-dependent, multi-session and session-independent tests on data from two session recorded with writer A. The results were acquired after 20 training iterations.

For the sessions with the old recording software, the first two iterations of the alphabet were skipped, since the test persons might not be used to the recording and writing procedure yet and thus write in an unusual way. Only single characters from these two iterations were used to replace erroneous characters in the further iterations. This means, if a writer made a fault, this was noted and data from the first two iterations was used instead. In the case of the newer software, people were asked to make themself acquainted with the glove and recording was started when the test person felt comfortable with the system. Since this software version allows easy repeating of characters, replacement was not necessary, every writer was able to correct himself in place. Either way, the test persons decided by themselfes, whether a character was erroneous or not. They were told to repeat characters in case of a wrong motion sequence. A wrong motion sequence is given, if the motion would not lead to the right capital block character on an imaginary blackboard or if the sequence of strokes is not consistent with the rest of the recordings. In case of an obvious erroneous manual segmentation, test persons were also asked to repeat the character. Again, I want to point out that the test persons were only told to write in capital block letters. There were no further constraints on the shape of the characters. As one can imagine, people do not write standarized block letters, everybody still has its individual way of writing. This was allowed for several reasons. First of all, people should write in a natural way, to get realistic data. This is not possible, when one has to concentrate on writing characters in an unusual way. Second, by observing the test persons, one can get an impression, which letters have a large variance in the way they are written and which are generally written in the same manner by all people. Of course, one has to be careful, when drawing conclusions based on an experiment with ten test persons, but a tendency can be identifiable.

5.4.2 Writer-Dependent

Writer A only

As for digit recognition, two sessions were recorded with writer A, the first one with 1300 characters, the second one with 650 characters. In the first session, 50 iterations of the alphabet in natural ordering were recorded. In the second session, the new recording software was introduced and 25 iterations of the alphabet were recorded. Experiments were performed on the session-dependent, session-independent and multi-session task. Table 5.4 shows the performed experiments and

Session	States	Gaussians per State	Gaussians Total	Recognition Rate
$A_{c,3}$	12	2	24	98.4
$B_{c,1}$	10	1	10	100.0
$C_{c,1}$	8	1	8	97.4
$D_{c,1}$	8	1	8	96.4
$E_{c,1}$	6	1	6	89.2
$F_{c,1}$	6	3	18	91.8
$G_{c,1}$	8	2	16	94.9
$H_{c,1}$	6	4	24	93.3
$I_{c,1}$	10	1	10	97.4
$J_{c,1}$	12	5	60	94.4
average				95.3

Table 5.5: Results of the writer-dependent task on the data of ten writers.

the corresponding results. For the session-dependent experiments and the multi-session experiments with the combined data from session 1 and 2, data was split into a training set (40%), a development set (30%) and a test set (30%). Different parameters for the number of Gaussians and states were tried. To be precise, a number of 6,8,10 and 12 HMM states and 1 to 5 Gaussians per state was probed. All possible combinations of parameteres were tried and in any case 20 training iterations were performed. The results of the session-dependent evaluation are comparable to the digit recognition results. For the session-independent evaluation, one session was taken as training set and the other session as test set. The optimal parameters from the session-dependent test were used for the number of HMM states and Gaussians per state. The session-independent result is lower compared to digit recognition. But in general, one can say, that performance is good in comparison to digit recognition. One has to take into account, that the number of possible classifications per written character increased from 10 to 26.

Writers A-J

A session-dependent recognizer was evaluated on the ten sessions from the writers A to J. The data of each writer was split into a training set (40%), a development set and a test set (30% each). Different numbers of HMM states and Gaussians were tried and the best performing system on the development set was chosen and evaluated on the test set. The results on the test set are given in table 5.5. For each test person, the respective best performing set of parameters was chosen. As a consequence, the results in the table are produced by different parameter settings. The table shows the different settings for the HMM topology and number of Gaussians, which resulted in the best recognition rates. The topoloy of the HMMs was varied in the number of states, more precisely 6, 8, 10 and 12 states were probed. For the Gaussian Mixture models, a number of 1, 2, 3, 4 and 5 Gaussians was used. Each combination of these parameters was used to train and test a recognizer. The parameter settings cannot be generalized to the writer-independent case. When dealing with different writers, the models have to cover more variants and varieties in writing and therefore, more complex models are needed. As our aim is to build

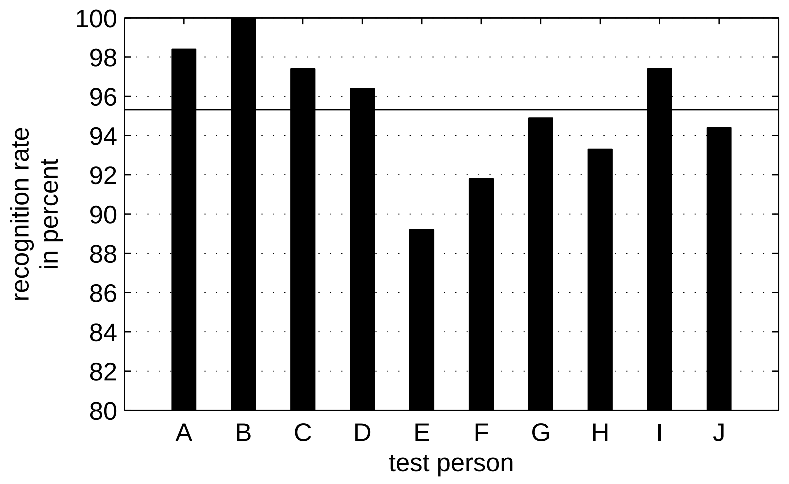


Figure 5.1: Recognition results of the writer-dependent evaluation. The line marks the average result over all test persons.

a writer-independent system, no further analysis to find an overall best parameter setting for the writer-dependent case was done.

Figure 5.1 shows all the results in a diagram. As one can see, the results on each of the individual writers are all between about 90 to 100 percent. The worst recognition rate was on data set H_1 with 89.2%. In general, one can say that recognition rates on writer-dependent systems are good. We take a closer look at the worst performing system to get an impression of typical misclassifications and problems. In absolute numbers, the recognition rate on the testset of writer E means, 21 hypothesis out of 195 were wrong. On the development set, a recognition rate of 91.3 was reached, which equates to 17 misclassifications out of 195 utterances. Figure 5.2 shows the combined confusion matrix for the best performing recognizer for data set E_1. This means the confusion matrices for the development set and test set were added together. The figure also shows the total number of false negatives and false positives for every character. Again the results of the development set and the test set have been added together. Clearly X and Y account for most of the confusions. P and D are also often confused. The confusion of X and Y and D and P is not surprising and probably a remaining challenge. Writer E wrote X and Y in the same style, i.e. both characters with the same stroke sequence and same stroke direction. For this writer, X and Y only differ in the length of the first stroke, which is in case of a Y approximately half the length of the corresponding stroke when writing an X. In case of D and P, the problem is similar to the one already observed for the digits 0 and 6. Again the lack of visual feedback might be a problem. The character I is misclassified four times in total, but its always misclassified as another character. This issue would need further investigation, there is no intuitive explanation. Other confusions in this particular result are also surprising, like for example T and M, but since they all occured only once, I did not analyze this any further. People weren't observed all the time, so there might be already errors in the data itself and there is no easy way to verify, what the test person actually wrote.

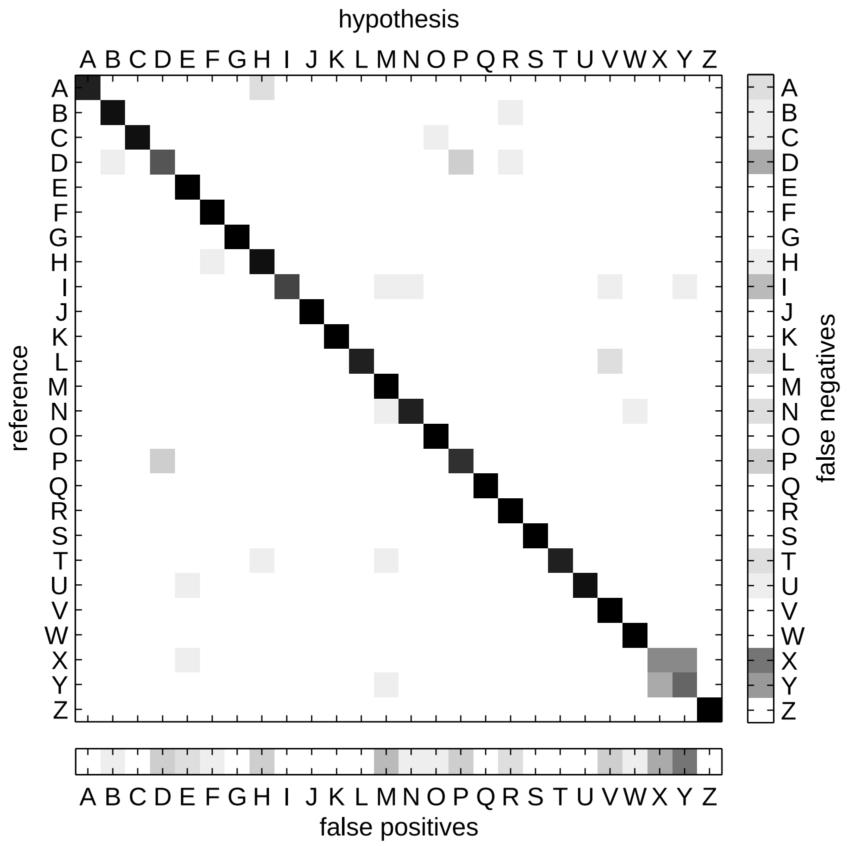


Figure 5.2: Confusion matrix of the writer-dependent evaluation of test person E. The matrix shows the combined results on the development and testset. The scale is the absolute number of classifications.

5.4.3 Writer-Independent

So far, we only examined writer-dependent systems. While the results were already promising, we cannot yet state any assumption on the performance of a general writer-independent system, i.e. a system evaluated on data of a writer, of whom no data is in the training set. To analyze this issue, the collected data from the writers A-J was used to examine performance of the recognizer for the writer-independent setting.

Cross Evaluation Set-Up

On the collected data, a leave-one-out cross validation was performed. The data from one test person was taken as test set and the remaining data from the other test persons was used for training. This was done for all ten, respectively nine test persons. Cross validation was performed one time on all data and one time on the data of right-handed test persons only. Because of being left-handed, test person C writes many characters in a different way, than a right-hander typically does. To exclude this effect, a system was trained using only the data from the nine right-handed writers. Both settings were evaluated once on accelerometer and gyroscope data and once on accelerometer data only, i.e. altogether, four different cross validation settings were evaluated. This was done to evaluate the possibilities of designing a device only using accelerometers. A device, which only relies on

Test Person	WI(LOOCV) all acc	WI(LOOCV) all acc+gyro	WI(LOOCV) only right-handed acc	WI(LOOCV) only right-handed acc+gyro
A	74.6	74.6	73.1	75.4
B	81.2	83.1	84.6	83.5
C	29.1	49.2		
D	81.9	90.2	82.6	90.5
E	76.3	85.9	76.6	86.9
F	73.9	76.1	73.4	77.2
G	77.9	80.5	78.2	80.6
H	66.9	73.9	69.4	72.9
I	82.2	87.1	81.9	86.0
J	75.1	83.9	76.5	84.3
av	71.9	78.5	77.4	81.9

Table 5.6: Results of the writer-independent recognition task. Each row corresponds to one test person. The second and third column show the results of training and testing on all writers including the left-hander, whereas the fourth and fifth column show the results of training and testing on right-hander data only.

Data	Sensor Channels	HMM States	Gaussians per State	Gaussians Total	Iterations
all (A-J)	a_{xyz}, g_{xyz}	12	6	72	10
all (A-J)	a_{xyz}	12	4	48	10
right-handers (A,B,D-J)	a_{xyz}, g_{xyz}	15	6	90	10
right-handers (A,B,D-J)	a_{xyz}	12	5	60	10

Table 5.7: Best performing overall parameter settings for the writer-independent experiments

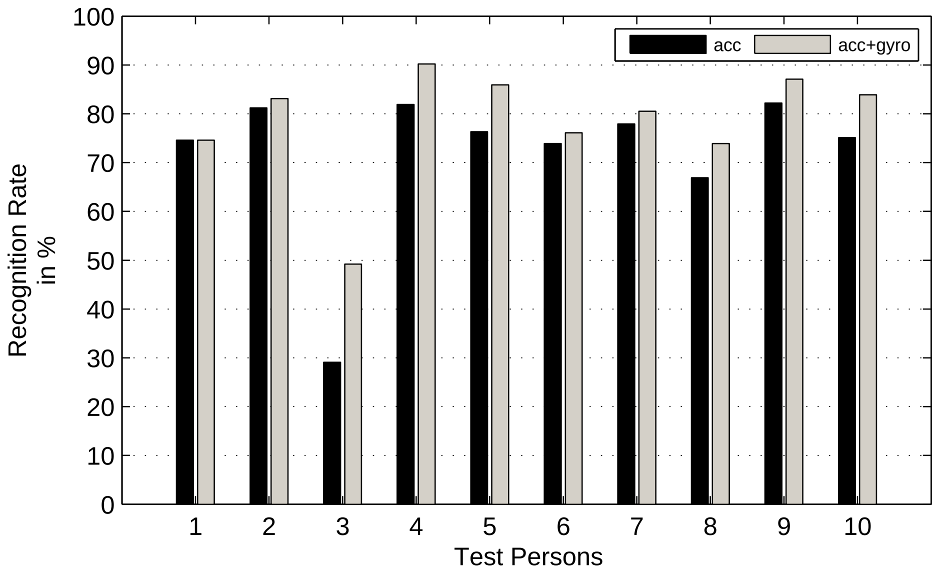


Figure 5.3: Results of the leave-one-out cross validation on all test persons, including the left-hander. The results are shown for the accelerometer and gyroscope evaluation, as well as for the accelerometer only evaluation.

accelerometers would be smaller, especially concerning the height of the sensor, it basically would be flat. If the recognition performance on accelerometer only data is good, this could motivate the construction of an even smaller and more unobtrusive device. Table 5.6 lists the results of the experiments.

Though still using a small data collection with few test persons, one should get an impression, how well the trained models generalize on data of arbitrary new writers. The used parameters for the number of HMM states were 8, 10, 12 and 15. The number of Gaussians was varied from 2 to 6. All possible combination of parameters were tested and the recognizers were trained for 10 training iterations. Table 5.6 shows the results of all performed experiments. Every row is associated with one test person which means, this persons data was used as test set. The parameter set, which gives the best average performance for each of the four experiments, was chosen. That means, the average recognition result was maximized for constant parameters over all ten or respectively nine individual tests. The results are shown in the bottom row of table 5.6. The corresponding parameters for the number of states and Gaussians are shown in Table 5.7.

Problems with the Left-Hander Data

Figure 5.3 shows the results of the cross validation, including all ten test persons. When looking at the results, first thing, one notices is the low recognition rate on the data of test person C. This is the left-handed person. As we can see in table 5.5, this writer has a very good writer-dependent result of 97.4%. So there is no argue about the quality of the data itself. The writer seems to have written in a very constant way and also might have an individual writing style with low ambiguity between characters. It can be assumed that the main reason for the poor result in the writer-independent test is the left-handed writing of test person C. From observation, it is known, that one very obvious difference to all other writers exists. Almost all

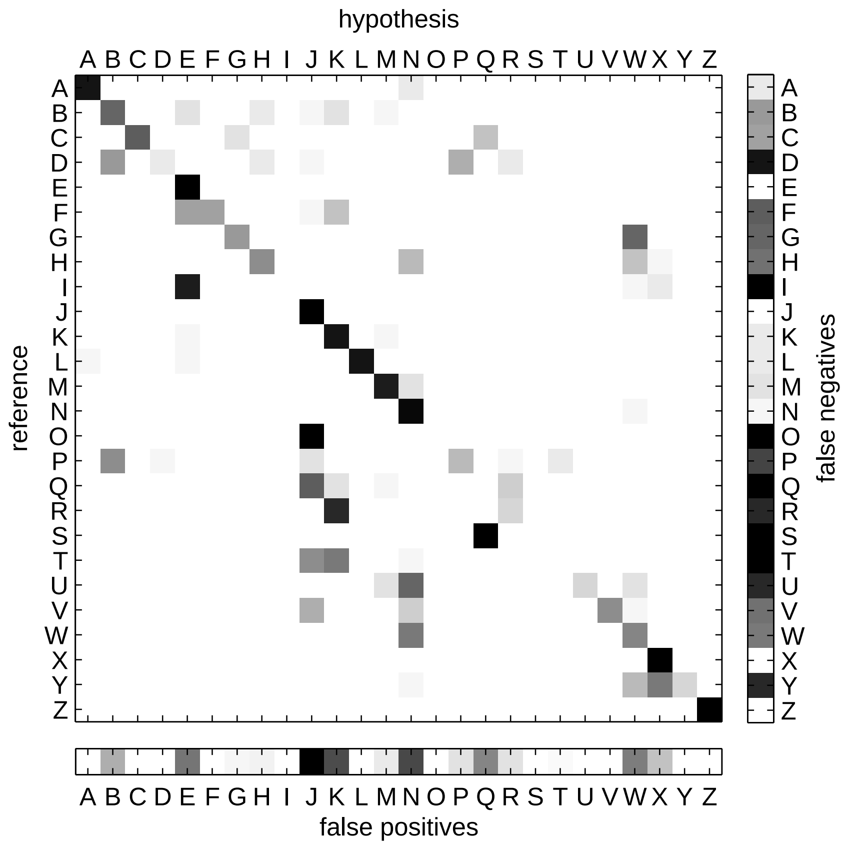


Figure 5.4: Confusion matrix of the evaluation on the left-hand written data set.

horizontal strokes were done from right to left by test person C, whereas right-handed writers generally seem to make isolated horizontal strokes from left to right. Test person C did write the characters A,E,F,H,I,T,Z with horizontal strokes from right to left. Additionally, the circular motion of the characters O and Q was written clockwise, whereas all right-handed writers do it counter-clockwise. That means, considering only this obvious difference, already 35% of the characters are done differently by test person C. It seems reasonable to assume, this is typical for left-hand writing, but for sure, more than one left-handed test person must be observed to verify this assumption. Figure 5.4 shows the corresponding confusion matrix. When looking at the already identified problematic characters, the characters I, T, O and Q are in fact rarely classified correctly. Also F and H are often misclassified. Finally A, E and Z together account for only two errors in total. Furthermore, it is surprising, that S is always recognized as Q, R almost always as K and G very often as W. So, we can clearly see the effects of left-hand writing, but also, the writer seems to have some individual writing specialities, which lead to other confusions. In fact, this test person had a very swinging writing style compared to the other test persons. The resulting effects are hard to analyze, since only the sequence of strokes was observed and documented.

General Problems of Writer-Independent Recognition

Figure 5.5 shows the recognition results for the writer-dependent and the writer-independent evaluations with accelerometer and gyroscope data. As one can expect,

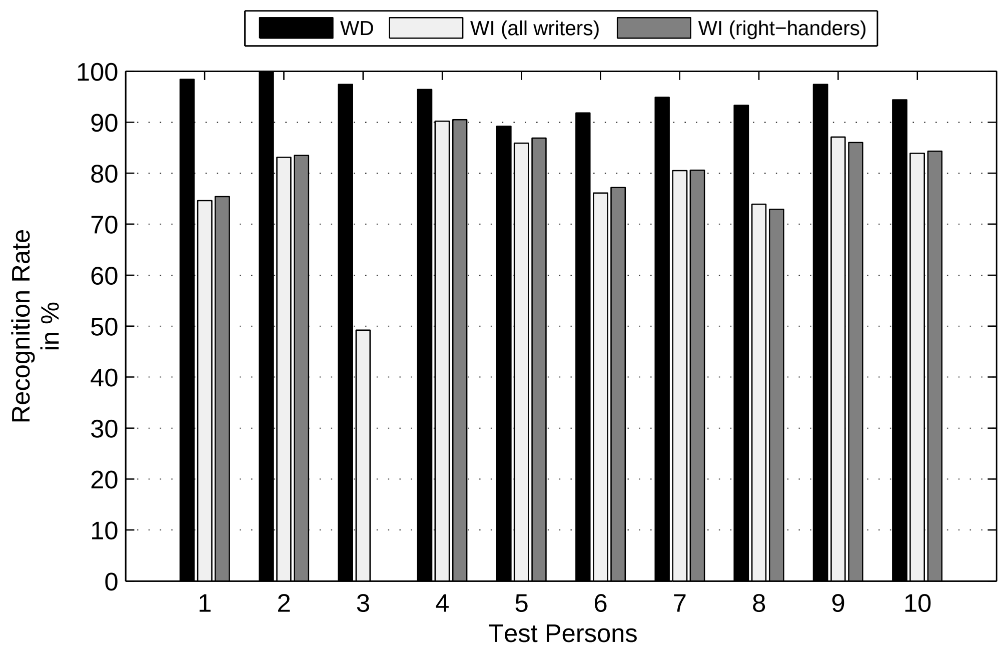


Figure 5.5: Results of the writer-dependent and writer-independent evaluations. The writer-independent results are shown for the evaluation with all ten writers and for the one with only right-handers.

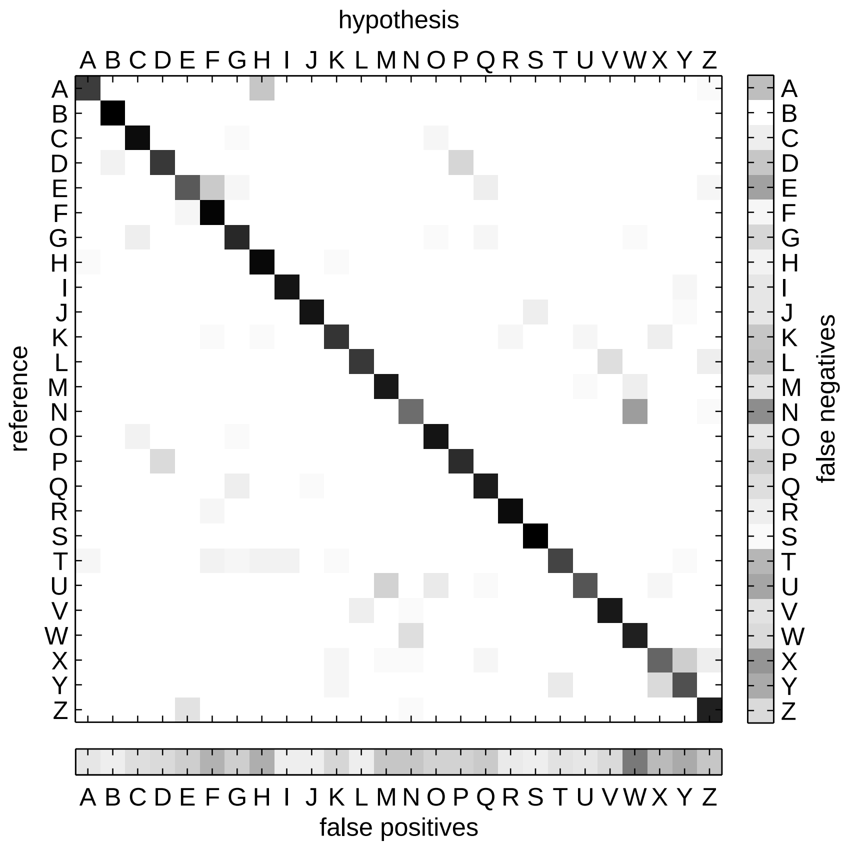


Figure 5.6: Accumulated confusions for the cross validation of the right-handed writers. Rows are references and columns are hypothesis.

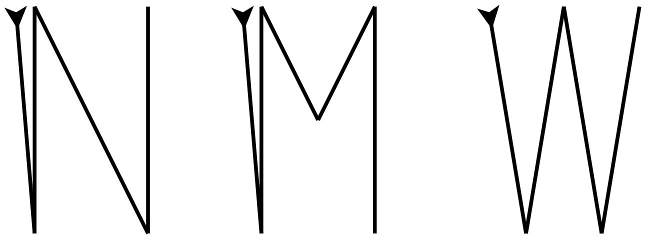


Figure 5.7: This figure illustrates observed writing variants of N and M. In case of the character N, this results in an ambiguity with the character W.

overall average recognition rates are slightly better, if train and test is performed on right-hander data only. But this is mainly due to the absence of the low recognition rate from the left-handed writer. The existence of the left-handed data in the training set does not seem to lower the overall recognition performance. A possible reason for this might be the high number of parameters, especially Gaussians per state. The models can then adapt to the left-hander writing variants without loosing significant performance on the other variants. For some test persons, the performance even got worse, when leaving out left-handed training data. For these persons, there might have been writing similarities with the left-hander and therefore the recognizer benefits from the additional training data.

In general, the overall performance of the writer-independent system is significantly lower than the writer-dependent system. This is not surprising, since the task is harder, but the main reasons should be investigated. For this analysis, I will consider the right-handed writer system only, to exclude effects caused by the special differences when writing with the left hand. Figure 5.6 shows the accumulated confusion matrix of the cross validation, that means confusion matrices of the test on the individual writers were added together. As one can clearly see, certain pairs of characters are subject to frequent confusion. First of all, the pair (N,W) was misclassified most frequently. Alltogether 116 misclassifications occured for those two characters. The reason for this confusion gets obvious, when considering the way, N and W were written by the test persons. At least three test persons wrote the N exactly like a W, because they started the N in the upper left corner and the movement to the real starting point of the N in the lower left corner was already part of the writing. This particular motion is also often seen, when people write an M, but this does not lead to an ambiguity. Figure 5.7 illustrates the special writing variants of N and M compared to W. The similarity between N and W has a major negative impact on the recognition performance. If one writes an N exactly like a W, clearly the probability of recognizing a W and thereby misclassifying it, is high. But additionally, during training the recognizer gets all these W-like N's labeled as N and adapts its models to this variant of motion. As result, the recognizer also classifies W's as N, when tested. On the isolated character level, there is not much we can do about it. If the stroke sequence is exactly the same, there is few difference to recognize.

A more obvious ambiguity than the one between N and W is the one between P and D, which accounts for 70 misclassifications in total. Here, definitely the stroke sequence is the same and the characters only differ in the endpoint of the second stroke relative to the first one. The resulting difference in the size of the second stroke, which should have approximately double the height than the one of P, is

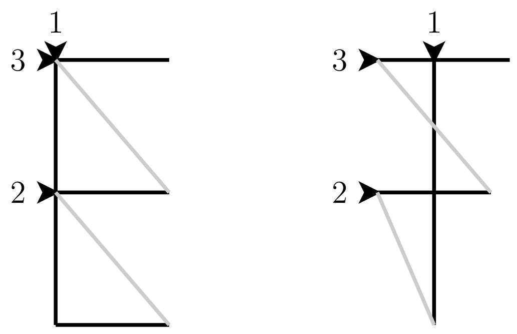


Figure 5.8: Similarity between E and F. Grey strokes would be pen up motions on a real blackboard.

hard to discriminate based on acceleration and angular velocity signals. This is the same kind of ambiguity, like the one, between the digits 0 and 6, which we already discussed in section 5.3.2. I again repeat my opinion, that to some extent, the reason for this particular kind of misclassification is the absence of visual feedback. There are 225 recordings of each grapheme in the dataset, that means 450 D's and P's together. The recognizer is still a lot better than guessing, even on such hard to discriminate single characters. The pair of X and Y also belongs into this category of non surprising misclassifications. Whereas the confusion between E and F does surprise at first sight. Similarities surely exist, but still the F has one stroke less, which makes printed E's and F's easy to discriminate. In this case, it is important to remember the fact, that our signals represent the whole motion, not only pen-down movements. Figure 5.8 shows possible writing variants of E and F observed in the test persons, including the motion between the actual strokes. One can see, that the whole stroke sequence for both characters is the same, except one stroke, which is missing for the F. Considering that people do not write exact strokes, there might even be more similarity, since writers might do a short motion to the right after the first vertical stroke. In this particular case, it should be noted, that the number of confusions is very imbalanced, an E was recognized 46 times as F, and an F was recognized 7 times as E. When looking at the individual confusion matrices of every writer shown in figure A.4 in appendix A.2, one can see, that this particular confusion only appears for the writers B, F and H. These are also the only writers, that do write an E in the stroke sequence shown in figure 5.8. We have already seen, that there are four writing variants of E in the data set, whereas F is basically written the same way by every writer. The HMM for E has to model four different motion sequences and thus has to cover a large variance of observation sequences. As result the model for F might produce a higher probability on the three writers that do write their E similar to an F.

Besides these intuitively understandable ambiguities, others do exist. The frequent misclassification of Z and E for example, cannot be explained by obvious similarities in shape or motion. Again, a closer look at the individual confusion matrices (see figure A.4 in appendix A.2) gives hints for possible reasons. This particular confusion only occurs for writer B and H. These are the only two writers, who write Z with a third horizontal stroke in the middle of the character. With this stroke, Z is more similar to E, but there is probably also another reason. When the data of B or H is used as test data, there remains only the data of one writer with this particular writing variant in the training set. Seven other writers contribute data for Z to the training set, who all do the Z in the "regular" way. By this, the major variant might be predominantly represented in the HMM model, leading to poor results for the special variant of Z. To sum up the analysis of the misclassifications, one can say that most of them do not surprise and seem to be the result of obvious ambiguities. Some are very clear, because of similarities in the shape of characters, and some get very clear, when taking the way, people actually write, into account. A possible approach to increase recognition performance would be the training of individual models for different writing variants of one and the same character. Based on the assumption, that an individual writer is generally consistent in the way he or she writes, a recognizer could adapt itself to the writer by choosing the appropriate models for the writing variants, which would be subject to further investigations.

5.5 Word Recognition

Since recognition results on single characters already showed good performance, a word recognizer was built. In section 4.5.1, it is outlined how such a recognizer could be implemented based on individual HMM models. The character models are concatenated to form models of words. This way, it is only necessary to have good models for the 26 characters of the alphabet and the repositioning movements between the characters. Although only a predefined vocabulary is recognizable, it can be easily expanded or changed, since new word models can be build from the existing set of character and repositioning models. I performed experiments on writer-dependent and writer-independent word recognition. For that reason 100 english words were recorded by test person A during one session. That means no results on session-independence and only very limited results on writer-independence can be obtained. But based on the results and the experience gained in character recognition experiments, the results on word recognition can be extrapolated and should give a good impression, on how hard this task is. In all experiments on word recognition, a 0-gram language model was used. That means every word from the vocabulary has an equal a priori probability and the context in which a word occurs is not taken into account. Higher order language models are succesfully used in speech and handwriting recognition. I therefore expect a performance boost for a possible future usage of such a model in this application as well.

Training of Repositioning Models

To train the repositioning models, the signals of the motion between writing two characters were recorded in the sessions $A_{c,1}, F_{c,1}, \cdots, J_{c,1}$. The other sessions were recorded with an earlier version of the recording software and therefore there was no recording of repositioning movements for the sessions $B_{c,1}, \cdots, E_{c,1}$. These repositioning motions are not exactly the same than the repositioning between characters when writing whole words, due to the stops for manual segmentation, but they should provide good initial parameter values for further training. In all experiments, I used one single HMM for the repositioning movement. One model was built for the writer-dependent case and one for the writer-independent case. I basically repeated the experiments on character recognition with an additional model for the repositioning. For the writer-dependent case, the dataset $A_{c,3}$ was used as train set and $A_{c,2}$ as development set. For the writer-independent case, the data sets $F_{c,1}, \cdots, J_{c,1}$ were used for training and $A_{c,3}$ was used as development set. For the

	Writer-Dependent		Writer-Independent	
	States	GMM	States	GMM
Characters (A-Z)	12	2	15	6
Repositioning	7	2	12	6

Table 5.8: The HMM topologies used for the word recognition experiments. All word models were built of HMMs with the specified topologies.

character models, the optimal topology and number of Gaussians from the experiments on character recognition was taken, that means 12 states and 2 Gaussians for the writer-dependent test, which was the optimum of the session-dependent test on session $A_{c,3}$ and 15 states and 6 Gaussians, which was the optimum for the right-hander writer-independent evaluation. For the repositioning model, I tried different parameter settings for the topology and the number of Gaussians. To be specific I used 3,5,7 and 12 states and 1 to 6 Gaussians per state. For the writer-dependent case, the repositioning model with 7 states and 2 Gaussians per state resulted in the highest recognition rate on the development set. For the writer-independent case the repositioning model with 12 states and 6 Gaussians resulted in the highest recognition rate. The repositioning models were trained using viterbi training with 10 training iterations before evaluation on the development set. Table 5.8 shows the resulting HMM topologies for the word recognition experiments.

Experiments

I already specified the topology and number of Gaussians used for the character and repositioning models. Besides that, I take two kinds of models for the experiments, which I denote by *k-means initialized* and *viterbi initialized. Viterbi inititalized* models are the character and repositioning models from the character recognition task after the training procedure, i.e. the HMMs have been initialized by clustering and afterwards been trained on character training data for several training iterations. *K-means initialized* models have not yet been trained on the training data but have been initialized by k-means clustering on the training data, i.e. they are already training data specific but no viterbi training to optimize them has been done. This was done to investigate, if the training on isolated characters leads to specialized models, which do not perform well when used for word recognition. For the writer-dependent test, character and repositioning models initialized on data of writer A were used. Word data was split into a training and a test set (50% each). The recognition performance was tested on the test set without any training on word data and after five training iterations on the training set. For the writer-independent test, character and repositioning models initialized on data of writers F to J were used and there was no additional training on the word data, since word recordings were only available from writer A. The same test set as for writer-dependent evaluation was used for the writer-independent case. Table 5.9 shows all performed experiments.

As vocabulary, a subset of a list of the top 1000 most frequent English words was used. The list was obtained from the University of Leipizg[2]. Such a list always depends on the analyzed sources and therefore there is nothing like "the 1000 most

[2]http://www.wortschatz.uni-leipzig.de/html/wliste.html

Task	Data		Character Model	Word Training	Recognition
	Train	Test	Initialization	Iterations	Rate
WI	-	$A_{w,1}$	k-means $(B_{c,1}, \cdots, J_{c,1})$	-	42.0
WI	-	$A_{w,1}$	viterbi $(B_{c,1}, \cdots, J_{c,1})$	-	82.0
WD	-	$A_{w,1}$	k-means $(A_{c,1})$	-	72.0
WD	-	$A_{w,1}$	viterbi $(A_{c,1})$	-	92.0
WD	$A_{w,1}$	$A_{w,1}$	k-means $(A_{c,1})$	5	96.0
WD	$A_{w,1}$	$A_{w,1}$	viterbi $(A_{c,1})$	5	96.0

Table 5.9: This table shows the results of the word recognition experiments.

frequent words in English". But this is not very important for our purpose. The use of this list should just guarantee to work with realistic data. A subset of 100 words was chosen, with words of length two to eleven characters. For every length, ten words were randomly selected from the list. This way, we have an equal distribution of words over the chosen length interval, which should ensure we get an impression, wether the length of words is a problem or not. The complete list is given in appendix A.3.

Table 5.9 shows the results of the experiments. Without any training on the recorded words, the writer-dependent system already reaches 92% for the *viterbit initialized* models. With additional training on words, the writer-dependent systems both reach 96%, no matter if models are initialized by *viterbi* or *k-means*. The writer-independent system reaches notable 82% with no training on words at all, but with writer-independent *viterbi initialized* models. Training on word recordings was not possible for this system, since there were no recordings of other writers. The results of the writer-independent test have to be interpreted carefully, since only one writers word recordings were available. To make general statements on writer-independent word recognition, one needs to record a larger dataset with more different writers. But the achieved results are very promising, especially concerning the fact, that no training on word data was done to reach this recognition rate. The writer-dependent results show that training on word recordings significantly raises the performance of the system and I would expect that word training would have a similar effect for a writer-independent system.

5.5.1 Live Recognizer

Motivated by the results of word recognition a live demo was built to demonstrate the abilities of the system. From a scientific point of view such a system can also be very useful. In a very intuitive way, one can experiment in different surroundings, different distances between sender and receiver and with different writing positions. So far the system seems to be reasonable stable, so this live demo can be seen as a proof of concept of actual practical implementation. The system runs on a laptop with an Intel Core 2 Duo T8300 processor with 2.4 GHz and 2 GB RAM. For the live system, the trained word recognizer from section 5.5 was taken and a very small vocabulary of nine words was used. The necessary words to write the two sentences "Welcome to the CSL" and "This is our airwriting system" were included. Figure 5.9 shows a screenshot from a demo video of the live system. The user is offered a simple graphical user interface, through which he can start and stop recording.

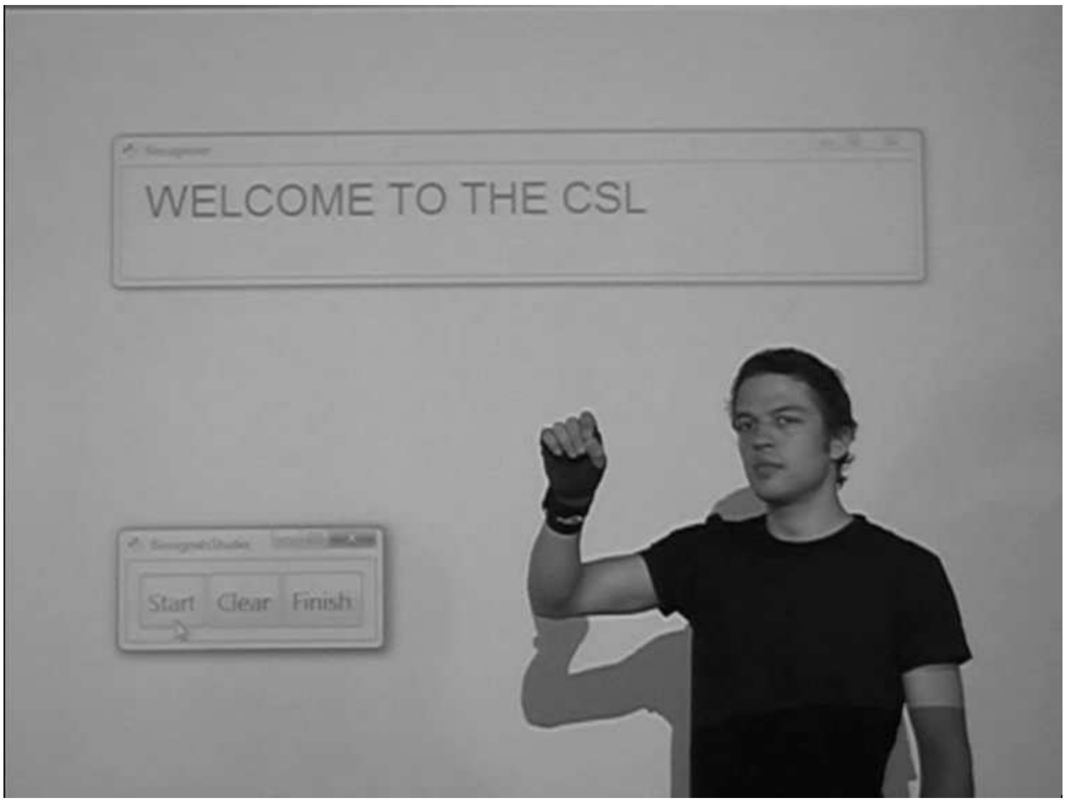


Figure 5.9: Screenshot from a demo video of the live recognizer. The data glove is worn at the right hand, the big window on the screen shows the recognized text and the small window has the control buttons to manually segment word boundaries.

Every recording is supposed to be a single word and the hypothesis is printed to the screen. Yet the system is writer-dependent, specifically it was trained on my own data. But the results of the writer-indepedendent character recognition and the writer-dependent word recognition systems lead to the assumption that a usable writer-independent online word recognizer is in reach.

6. Conclusion and future work

6.1 Summary

In my work I designed and implemented a data glove, based on inertial sensors. The data glove is equipped with three orthogonal gyroscopes and three orthogonal accelerometers to measure hand motion. Data is processed and sent to a computer via Bluetooth. The data glove is used to write in the air, like writing on an imaginary blackboard. Based on the signals the glove delivers, I developed an HMM based recognizer, using the existing Janus Recognition Toolkit. Several experiments were conducted to optimize and evaluate the system. Specifically, experiments were performed on single digit, single character and word recognition. For the task of character recognition, ten test persons contributed to the data collection. Writer dependent as well as writer-independent recognition was evaluated and arising problems were analyzed in detail. The writer-dependent recognition rate on single character recognition ranged between 89% and 100% for one of the ten test persons. The average recognition rate of writer-independent recognition was 81.9%. Based on a small vocabulary of 100 words, first experiments in word recognition were conducted. Finally, a real-time demonstration system was implemented, to show the functionality of the system in practice. There has already been some research in the field of airwriting recognition, using wireless, sensor equipped pens instead of a data glove. To my knowledge, this is the first work on expanding the recognition on whole words for the task of writing in the air.

6.2 Conclusions

From the experiments and real-time demonstration system, one can conclude that the implemented data glove works in a reliable and robust manner, considering data rate over the air, connection stability and quality of the sensor signals. Concerning the recognition, one can conclude, that continuous airwriting recognition is possible and within reach of current technology, at least for the writer-dependent case. High recognition rates on single digits and characters show, that the raw sensor signals are good features for recognition. The reconstruction of the trajectory is not necessary for the accomplished task but nevertheless would be helpful in many ways, for example for the analysis of misclassifications. The performed analysis shows, that problems arise, when the characters writing motions are very similar, which is not surprising. It is not yet clear, if the recognition performance can be increased significantly by improving the method or if this is mainly caused by the absence of visual feedback for the user, resulting in truly ambigious motions. Less intuitive types of misclassifications exist and are not yet understood. A deeper analysis of these problems should reveal possible solutions.

6.3 Future Work

The introduced system serves as proof of concept that handwriting recognition in the air with inertial sensors is possible. Future work would imply further miniaturization of the device and improvements of the recognizer.

On the hardware side, it would be desirable to have an even more unobtrusive device. This might be accomplished by abandoning of the gyroscopes, since their orthogonal alignment is the reason for the cubic shape of the sensor. Also, smaller radio transmission solutions are possible and power supply might be switched to lithium-ion batteries. This would make a redesign of the device necessary.

A further step in the recognition process would be the integration of a language model to move towards continuous handwriting recognition. A deeper analysis of the observed misclassifications is necessary to develop new and better features. The possibilities of individual models for writing variants should be evaluated, as well as context dependent repositioning models. Additionally, since writer-independence is a problem, the idea of a self adapting recognizer seems promising. The degrees of freedom of the writer should be increased, so that at least the unnatural constraint to write with fixed wrist can be removed. Going from capital block letters to unconstrained writing is desirable, but should be regarded as a hard problem.

For practical use, the computational effort should probably be lowered. For that reason, tests should be performed, if the sampling rate can be lowered without loosing recognition accuracy. Possibly, better features than raw sensor signals exist and would allow some compression, which would increase decoding speed. Efficiency is important, since in wearable computing, computational performance and energy is strongly bounded.

A. Appendix

A.1 Circuit Layout

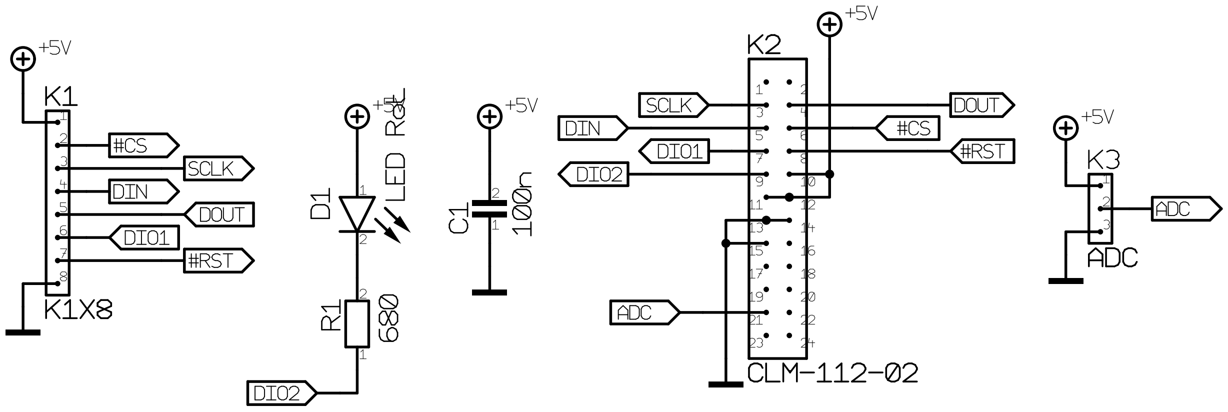


Figure A.1: Sensor board circuit diagram

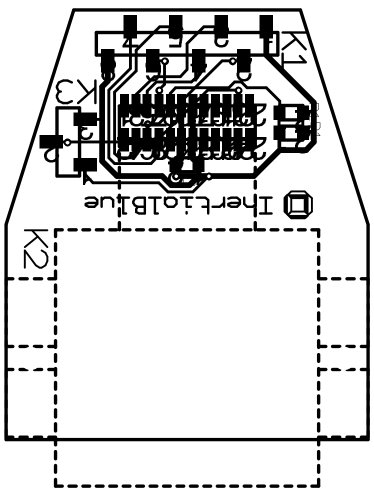


Figure A.2: Sensor board circuit layout

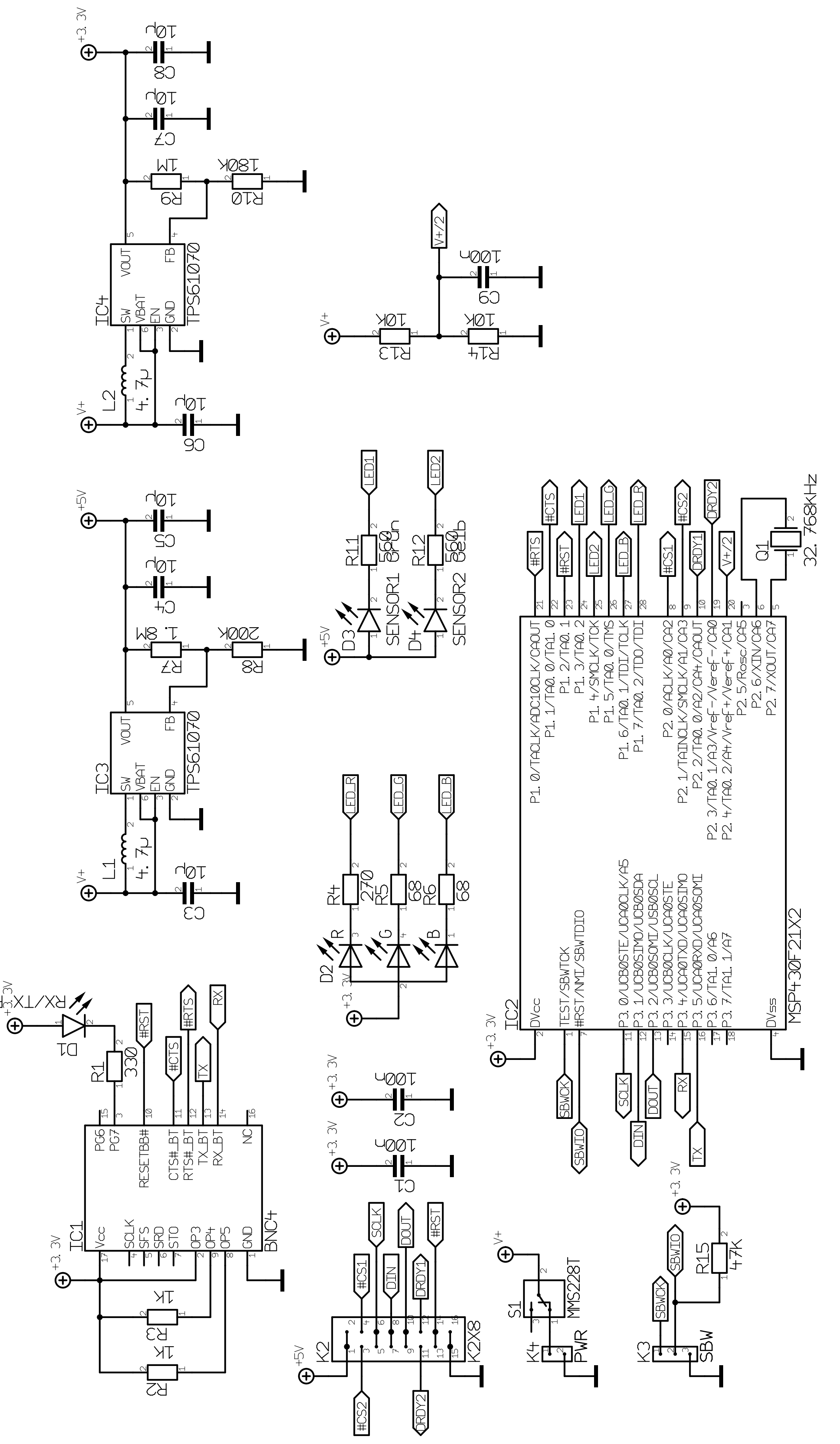


Figure A.3: Controler board circuit diagram

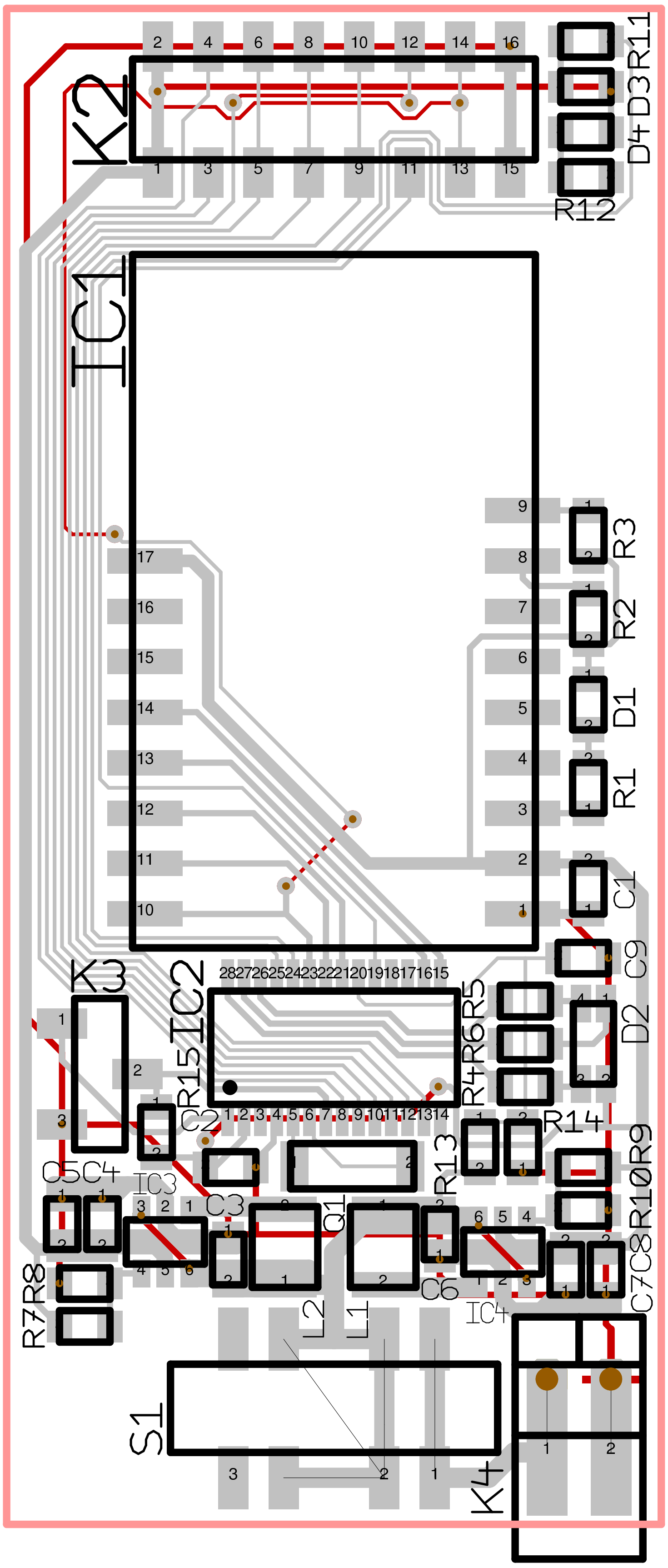
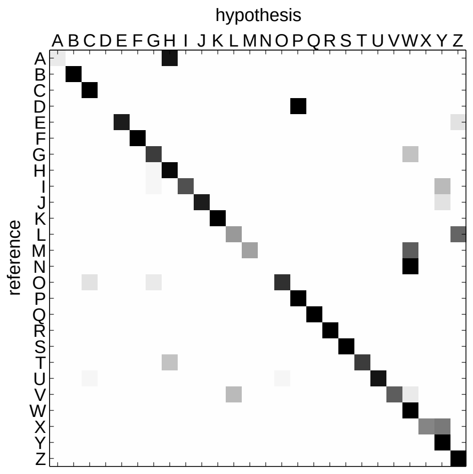
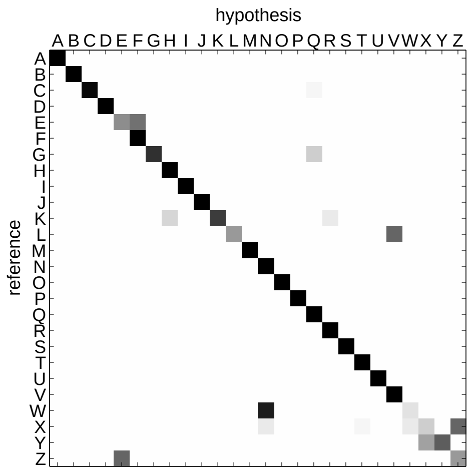


Figure A.4: Controller board circuit layout

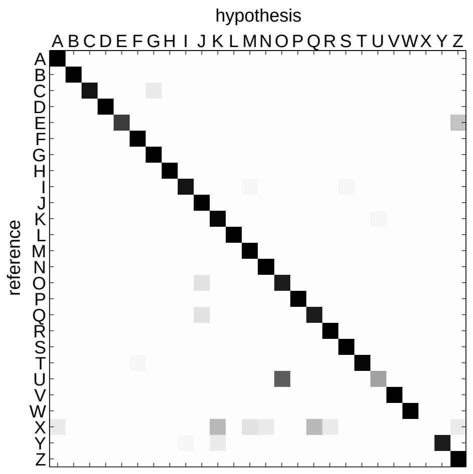
A.2 Confusion Matrices



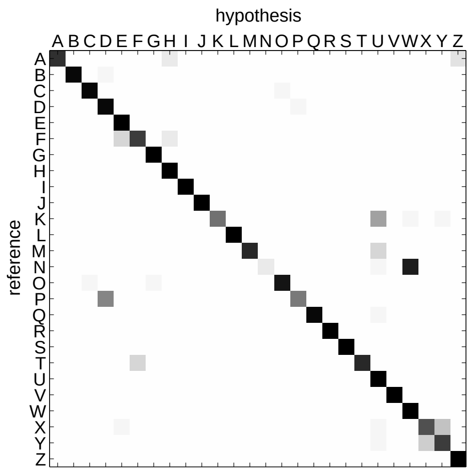
(a) Writer A



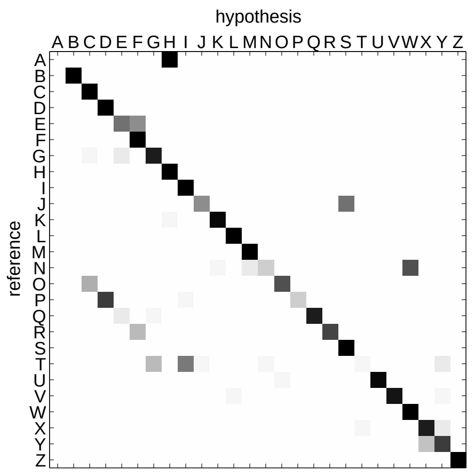
(b) Writer B



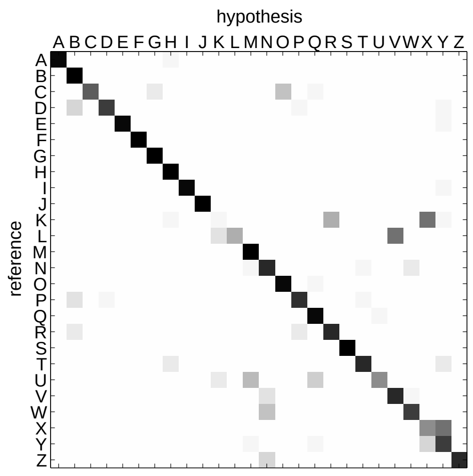
(c) Writer D



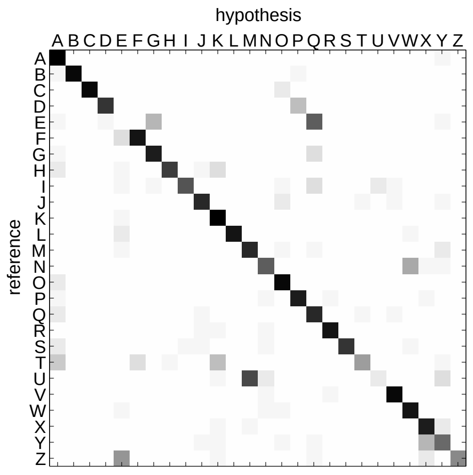
(d) Writer E



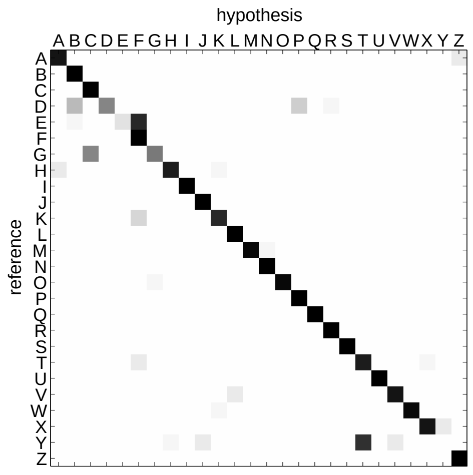
(e) Writer F



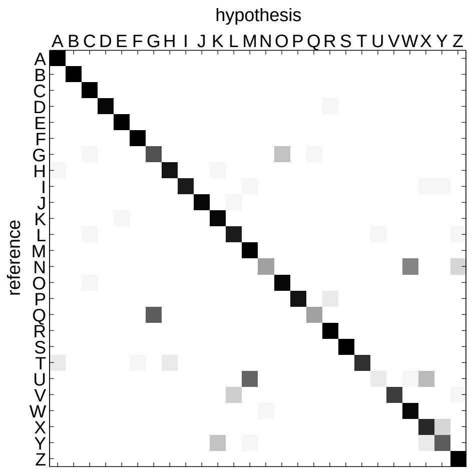
(f) Writer G



(g) Writer H



(h) Writer I



(i) Writer J

Figure A.4: Confusion matrices of the writers A-J for the cross-evaluation on the right-handed writers. The missing writer C is the left-hander.

A.3 Data sets

The following vocabulary has been used for the word recognizer:

1. of to in is on be by as at it
2. the and for was are has its not but can
3. that said with from have will were this more been
4. which would about their other after first years could there
5. system market people before should during number shares prices months
6. million company percent because program systems between billion through support
7. software business products computer american expected industry reported interest programs
8. companies officials president including available operating different according spokesman announced
9. government management technology department production washington operations investment processing additional
10. information development performance application environment significant association programming authorities

Bibliography

[AAS+09] O. Amft, R. Amstutz, Smailagic, A., D. Siewiorek, and G. Tröster, *Gesture-controlled user input to complete questionnaires on wrist-worn watches*, Human-Computer Interaction. Novel Interaction Methods and Techniques, Lecture Notes in Computer Science, vol. 5611, Springer Berlin / Heidelberg, 2009, pp. 131–140.

[BCK+03] W.-C. Bang, W. Chang, K.-H. Kang, E.-S. Choi, A. Potanin, and D.-Y. Kim, *Self-contained spatial input device for wearable computers*, Proc. Seventh IEEE International Symposium on Wearable Computers (2003), 26–34.

[BSLJ03] H. Brashear, T. Starner, P. Lukowicz, and H. Junker, *Using multiple sensors for mobile sign language recognition*, Proc. Seventh IEEE International Symposium on Wearable Computers, 2003, pp. 45–52.

[CGKP02] A.D. Cheok, K. Ganesh Kumar, and S. Prince, *Micro-accelerometer based hardware interfaces for wearable computer mixed reality applications*, Proc. Sixth International Symposium on Wearable Computers, 2002, pp. 223–230.

[CK04] S.-J. Cho and J.H. Kim, *Bayesian network modeling of strokes and their relationships for on-line handwriting recognition*, Pattern Recognition **37** (2004), no. 2, 253–264.

[CLL06] S.-D. Choi, A.S. Lee, and S.-Y. Lee, *On-line handwritten character recognition with 3d accelerometer*, Proc. IEEE International Conference on Information Acquisition, 2006, pp. 845–850.

[FGH+] M. Finke, P. Geutner, H. Hild, T. Kemp, K. Ries, and M. Westphal, *The karlsruhe-verbmobil speech recognition engine*, Icassp-97. **1**, 83–86.

[HHH98] F.G. Hofmann, P. Heyer, and G. Hommel, *Velocity profile based recognition of dynamic gestures with discrete hidden markov models*, Gesture and Sign Language in Human-Computer Interaction, Lecture Notes in Computer Science, vol. 1371, Springer Berlin / Heidelberg, 1998, pp. 81–95.

[HHK09] A. Hein, A. Hoffmeyer, and T. Kirste, *Utilizing an accelerometric bracelet for ubiquitous gesture-based interaction*, Universal Access in Human-Computer Interaction. Intelligent and Ubiquitous Interaction Environments, Lecture Notes in Computer Science, vol. 5615, Springer Berlin / Heidelberg, 2009, pp. 519–527.

[KC01] I.-C. Kim and S.-I. Chien, *Analysis of 3d hand trajectory gestures using stroke-based composite hidden markov models*, Applied Intelligence **15** (2001), no. 2, 131–143.

[KCK06] D.H. Kim, H.I. Choi, and J.H. Kim, *3d space handwriting recognition with ligature model*, Ubiquitous Computing Systems, Lecture Notes in Computer Science, vol. 4239/2006, Springer Berlin / Heidelberg, 2006, pp. 41–56.

[LO98] R.-H. Liang and M. Ouhyoung, *A real-time continuous gesture recognition system for sign language*, Proc. Third IEEE International Conference on Automatic Face and Gesture Recognition, 1998, pp. 558–567.

[MHRS$^+$04] R.M. McGuire, J. Hernandez-Rebollar, T. Starner, V. Henderson, H. Brashear, and D.S. Ross, *Towards a one-way american sign language translator*, Proc. Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004, pp. 620–625.

[OCB$^+$04] J.K. Oh, S.-J. Cho, W.-C. Bang, W. Chang, E. Choi, J. Yang, J. Cho, and D.Y. Kim, *Inertial sensor based recognition of 3-d character gestures with an ensemble classifiers*, Proc. Ninth International Workshop on Frontiers in Handwriting Recognition (2004), 112–117.

[OP07] I. Oakley and J.-S. Park, *Designing eyes-free interaction*, Haptic and Audio Interaction Design, Lecture Notes in Computer Science, vol. 4813, Springer Berlin / Heidelberg, 2007, pp. 121–132.

[PS00] R. Plamondon and S.N. Srihari, *Online and off-line handwriting recognition: a comprehensive survey*, IEEE Transactions on Pattern Analysis and Machine Intelligence **22** (2000), no. 1, 63–84.

[Rab89] L.R. Rabiner, *A tutorial on hidden markov models and selected applications in speech recognition*, Proceedings of the IEEE, vol. 77, 1989, pp. 257–286.

[RBA08] M. Rehm, N. Bee, and E. André, *Wave like an egyptian: accelerometer based gesture recognition for culture specific interactions*, Proc. of the 22nd British HCI Group Annual Conference on HCI (Swinton, UK, UK), British Computer Society, 2008, pp. 13–22.

[SMFW01] H. Soltau, F. Metze, C. Fügen, and A. Waibel, *A one-pass decoder based on polymorphic linguistic context assignment*, Asru (2001), 214–217.

[Sow08] T. Sowa, *The recognition and comprehension of hand gestures – a review and research agenda*, Modeling Communication with Robots and Virtual Humans, Lecture Notes in Computer Science, vol. 4930, Springer Berlin / Heidelberg, 2008, pp. 38–56.

[SPHB08] T. Schlömer, B. Poppinga, N. Henze, and S. Boll, *Gesture recognition with a wii controller*, Proc. of the 2nd international conference on Tangible and embedded interaction, ACM, 2008, pp. 11–14.

[Woo07] Oliver J. Woodman, *An introduction to inertial navigation*, Technical Report 696, University of Cambridge, 2007.

[WPZ$^+$09] J. Wu, G. Pan, D. Zhang, G. Qi, and S. Li, *Gesture recognition with a 3-d accelerometer*, Ubiquitous Intelligence and Computing, Lecture Notes in Computer Science, vol. 5585, Springer Berlin / Heidelberg, 2009, pp. 25–38.

[ZDLK08] S. Zhou, Z. Dong, W.J. Li, and C.P. Kwong, *Hand-written character recognition using mems motion sensing technology*, Proc. IEEE/ASME International Conference on Advanced Intelligent Mechatronics, 2008, pp. 1418–1423.