Signature verification in consignment notes

Riccardo Simon Corradin

Master thesis

Signature verification in consignment notes

Riccardo Simon Corradin

Master thesis

VU University
Faculty of Exact Sciences
Department of Computer Science
De Boelelaan 1081a
1081 HV Amsterdam

Supervisors: Dr. A.P.W. Eliëns. Co-supervisor: Dr. W.J. Kowalczyk

Host organization:
Logica Netherlands BV
Prof. W.H. Keesomlaan 14
1183 DJ Amstelveen
PO Box 159
1180 AD Amstelveen

Supervisors: Dr. Ir. H. van Thienen and Ms. I. van Zaanen

August 2008

Version Control

Version	Description
v 0.1	Draft version
v 0.2	Initial changes based on feedback by Ms. I van Zaanen and Ing. H. Brouwer
v 0.3	Made various medium changes to the document, based on feedback of University
v0.4a	Corrected version of never published v0.4. Concept version of thesis
v0.5a	Corrected version based on feedback of Dr. A.P.W. Eliëns and Dr. W.J. Kowalczyk.
v0.6	Corrected version based on feedback of Dr. Ir. H. van Thienen
v0.7	Added front page and finalized layout
v1.0	Added some minor correction based on the feedback of Dr. W.J.Kowalczyk

Table I: Version control

Preface

A large amount of time and effort is spent in realizing this thesis. The work that led to the current state of the thesis could not be realized without the help of certain people. In this section I would like to thank those persons.

First of all I would like to thank my supervisor Dr. A.P.W. Eliëns and co-supervisor Dr. W.J. Kowalczyk at the Vrije Universiteit for their high quality immediate feedback and assistance during the internship at Logica. Both showed great interest and dedication to this project.
Secondly, my sincere thanks to the coordinators at Logica Dr. Ir. H. van Thienen (who shared his experience in the field of academic research) and Ms. I. van Zaanen (who offered her expertise to get acquainted with Logica as a company). They both showed a high degree of interest, patience and availability during my internship.
Thirdly, I would like to thank Ing. H. Brouwer and Ir. C. Liefting, who both, unfortunately, left the Working Tomorrow department approximately two months after the initiation of this project. Despite this, they helped me to a great extend in defining the assignment and initiating the research.

Furthermore I would like to thank my Mentor, Mr. N. Elzer and my manager Mr. M. van Hilten for their interest and, more particular, the good conversations we had regarding the project and Logica.
Ing. C. Kooijman played an important role as well during early phases of my project and I would like to thank him for all his advice and patience. Although he is rather new as an employee at Logica, his presence was valuable and led to great improvements in the definition of the project.
In addition, my sincere thanks to Mr. M. Perdeck, who shared his expertise in the field of formulating documents and theses.

Finally, I would like to thank my girlfriend, family and fellow internship students for all their patience, support and input.

Abstract

Signature verification is a topic that, during the years, has not lost its relevance. In fact, it has become more popular in many different areas. Although less accurate than biometric techniques, such as fingerprint and iris scan verification, signature verification requires relatively less complicated and less expensive hardware to be realized. Currently, development within the field of signature verification is growing ([BioWeb, 1997]), as traditional algorithms are improved and new algorithms are created to compete with the traditional ones. According to [Gupta and Joyce, 2007], there is a considerable interest for reliable signature verification techniques. In addition, a method to automate this verification can reduce costs and accelerate processes within different organization.
In this thesis, different signature verification techniques will be studied. Out of these techniques a traditional and a newer technique are chosen, implemented and compared against each other. It should be emphasized here that on-line signature verification is used, in which dynamic information about a signature is known. This research will be focused on the realization of signature verification within transport companies, but the suggested techniques can be applied to any environment in which a digital pen and paper is used and the demand of signature verification is high.

Contents

1 Introduction

In the past couple of years, major changes (such as automated order processing systems) have been made regarding automation in the transport industry. However, a more particular aspect, the consignment note (regarding the delivery of goods) has not been fully automated yet. One particular technology is on its way to realize this. We are referring to the verification of handwritten signatures.
This thesis consists of research in verification techniques and their reliability and is realized in the form of a prototype. Signature verification (or authentication) is the process of verifying one input of a person against one input in the database. I.e.: the system verifies if the person is who he/she claims to be. Another widely used term is signature identification, which is the process of finding the person among other persons in the database, based on one input signature.
As can be seen, the two terms relate to each other, but their definition is different. In this thesis, signature verification is studied and realized, since the purpose is to verify signatures based on the input of different persons. However, the additional option to identify users, based on their signature, is discussed as well.

Up until now, a lot of development has been done. The main challenge in this project is to study different signature verification techniques and choose a small subset of them of which it is believed they perform well. These techniques are next applied and tested in a practical environment. The hardware to capture the signatures is a digital pen, which is able to register different signature data, such as the coordinates of a signature. For the selection of the verification techniques, different aspects are taken into account as is explained in subsequent chapters.

This research thesis starts with a problem statement. In this problem statement, the issue of realizing signature verification in consignment notes is analyzed and described. In addition, research questions are formulated, which play a key role in this research.

Next, Anoto technology (which enables a digital pen to store dynamic information about a signature) is highlighted and explained, since understanding this technology plays a major role in knowing how to couple the verification techniques to Anoto technology.

After the introduction of Anoto technology, an analysis of different signature verification techniques that have been used worldwide is made. From the analysis a small subset is derived. This small subset is then further analyzed and details of the techniques pertaining to this subset appear. These details are needed for the implementation of the techniques in Anoto environment.

Once the signature verification techniques are implemented and the results are gathered, an evaluation will be given. This evaluation will contain information about the used parameters and which combination of parameters yields the desired results.

The last phase of the project is improving (if this is possible) the implemented techniques to yield a higher accuracy. Since a lot of time is devoted to selecting signature verification techniques and applying these techniques in Anoto environment, little space is left for the practical realization of improvement. Therefore, advice is given of how to improve the chosen techniques, based on the experience gained during this project.

Finally, a conclusion is given, which takes into account the different phases of the project and contains statements about the reliability of the chosen techniques and Anoto technology.

2 Organization

In this chapter, a short overview of the company where this research has been done is given.

2.1 Logica

This research will be executed at Logica plc. in Amstelveen. The name “LogicaCMG” has been changed to “Logica” as of 27-02-2008. LogicaCMG plc. has been founded on December 30, 2002 from the former Logica plc. (60%) and CMG plc. (40%). Both ICT-service providers are originally English, but CMG was larger in the Netherlands than in the United Kingdom. Since January 13, 2006 the French Unilog company and, later that year, the Swedish company WM-Data joined Logica.

Logica is a leading IT and business services company, employing 39,000 people across 36 countries. It belongs to the top-20 of international ICT-service providers, regarding turnover. The turnover from traditional IT-services companies is mainly obtained from Europe and Australia. The turnover in the rest of the world is strongly bound to the telecommunication-industry.
It provides business consulting, systems integration, and IT and business process outsourcing services. Logica works closely with its customers to release their potential – enabling change that increases their efficiency, accelerates growth and manages risk. It applies its deep industry knowledge, technical excellence and global delivery expertise to help its customers build leadership positions in their markets. Logica is listed on both the London Stock Exchange and Euronext (Amsterdam) (LSE: LOG; Euronext: LOG).

Logica is characterized by the following mission statement:

“*To help leading organizations worldwide achieve their business objectives through the innovative delivery of information technology and business process solutions*”[1].

[1] Source: www.logica.com

2.2 Working Tomorrow

The research will be executed at the Working Tomorrow department. This program is started by Logica by the divisions EUT (Energy, Utilities & Telecom) and IDT (Industry, Distribution & Transport). Working Tomorrow employs students at different locations to work on their final thesis. All of the projects are innovative in the field of technology, concept or method, such as the development of digital paper and intelligent current.

The Working Tomorrow program has 3 main objectives:

- Recruit future employees
- Increase the reputation of Logica concerning innovation
- Use demos and prototypes in the negotiation trajectory of larger projects

This project is placed in the division IDT. In this division, several specializations (Business Units or BU's) are distinguished. In this situation, the student will be employed in the Business Unit "Competence Microsoft". The following figure shows the organizational chart of Logica and Working Tomorrow.

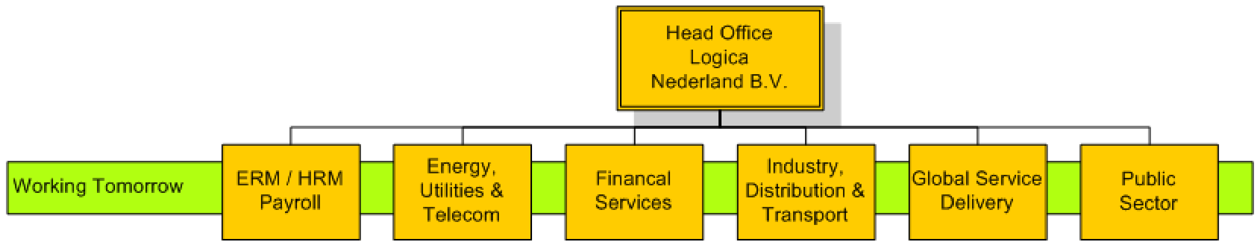

Head Office Logica Nederland B.V.
Working Tomorrow
ERM / HRM Payroll
Energy, Utilities & Telecom
Financal Services
Industry, Distribution & Transport
Global Service Delivery
Public Sector

Figure 2-1: Organizational chart of Logica Netherlands and Working Tomorrow

3 Problem statement

Anoto technology (which was mentioned in the introduction) allows the capturing and processing of handwritten forms. However, Anoto technology does not provide any further functionality, such as handwriting recognition and signature verification.

In transport companies, Anoto technology can be used to digitize the consignment notes. Consignment notes consist mainly of the signatures of the supplier, receiver and truck driver. Possibly some extra information regarding the contents of the cargo can be supplied (e.g.: if the count does not match or some of the cargo is damaged).

In some cases, handwriting recognition could be desired by those companies. The focus in this thesis will lie on signature verification, since handwriting recognition is a complete different topic.

When signing a consignment note, it is difficult to tell if the signature has been drawn by an authorized person and usually acceptance is made quickly (a cross is drawn or some other type of image which does not correspond to a signature). In order to solve this problem, a mechanism is needed that is reliable enough to verify a person's signature.

The problem statement can be formulated in short as follows:
Provide a solution to verify signatures in a reliable way, using Anoto technology.

By "reliable" it is meant at least better than guessing (50%). Of course the ideal goal is to yield a reliability of 99.9% or higher, but this is almost impossible to achieve, using existing techniques. Depending on the research that is done and the outcomes that are yielded, reliability will be defined.

The problem statement can be seen as the main goal of this research and as a beacon that will provide guidance in the right direction if we stray too much from our path.

3.1 Research questions

To approach the problem statement and initiate research, several questions need to be posed that help define the thesis and show in which areas the research is active. These questions are listed below.

1. Which techniques are available to realize signature verification?

   Different techniques have been developed using algorithms for classification, each presenting varying accuracy results. Classification algorithms may perform well on one signature dataset, but worse on another.
   These techniques have to be studied in order to compare them with each other. The most promising ones will be chosen for the experiment phase in this project.

   In the current context, a technique has to be found that performs well, both in terms of accuracy and integration with a practical environment. The environment in this project is characterized by Anoto technology. This means that it should be possible to couple the verification techniques to this technology.

   At this stage it is not possible to test the available techniques. Therefore an estimation and selection of techniques that are known to be fairly reliable, will be made.

2. Which of the available techniques should be selected for further analysis and what is the motivation behind this selection?

   After the analysis of available techniques, two are selected for further analysis. At this stage a study in which areas these techniques have been used and to what extent there accuracy is high is done. To develop the prototype, good knowledge of the reliability of the techniques is needed in order to give predictions on how accurate the techniques will perform on the prototype. Next to accuracy, other selection criteria such as speed and compatibility will be considered as well. To obtain all this information, the conditions under which the techniques are tested are inspected carefully.

   It will save a lot of time when the decision of chosen techniques has been thought over twice as this will reduce the possibility of choosing inappropriate techniques.

3. How can these techniques be implemented and evaluated?

   After having carefully selected techniques, they have to be coupled to Anoto technology in order to evaluate the results of these techniques. This may be the most challenging task in the project, since acquiring readable data from Anoto enabled hardware requires a license.

   A mechanism is needed that extracts readable data from the signatures and uses this as an input for the chosen verification techniques. This stage will consist of implementing an application, making connections between Anoto technology and this application and collecting the results.

4. How can the evaluated techniques be improved?

   During the evaluation phase of the techniques, methods to increase performance may be found. The real challenge in this phase is to try finding a general solution that will increase overall accuracy. In order to avoid a lot of trial and error, first a theoretical starting point has to be set up, which describes in what way accuracy could be improved. The actual practical implementation of improving the techniques is not executed but left open for future research.

5. Are the results from the evaluated techniques accurate enough to realize a reliable signature verification application in a real world environment?

   In the conclusion, based on the results of the previous stages, it will become clear if the evaluated results are adequate to provide users with accurate signature verification. I.e.: An answer to the problem statement will be given.

As can be seen, research questions can clarify a lot more than the initial problem description and gives a more detailed view of the project.

4 Previous work

From previous sections it can be concluded that signature verification is not a straightforward problem to solve. In this chapter a description of some earlier work and the introduction of common terms within signature verification are given.

Earlier, it was mentioned that a lot of research has already been done in the field of signature verification. On one side, techniques have to be defined and on the other side the digital pen environment has to be discussed.
An extensive table of on-line signature verification techniques is shown in Appendix A. This table is the same table as displayed in [Plamondon and Lorette, 1989] and is added here for clarity and completeness.
As can be seen a lot of different datasets and comparison methods have been used. In addition, different features have been selected according to the experiment being done. Furthermore, the size of the signature datasets also differs from one experiment to another. This makes it hard to compare the different experiment outcomes in a fair way.

Since the table dates from 1988, more recent techniques are not visible. These recent techniques are proven to be very powerful, since they use a larger dataset and provide a better classification.
These techniques are the following:

- Gaussian Mixture Models (GMMs)
- Dynamic Time Warping (DTW)
- Extreme points warping (EPW)
- Hidden Markov Models (HMMs)
- Artificial Neural Networks (ANNs)

Gaussian mixture models are discussed in [Richiardi and Drygajlo, 2003], in which these models are used for classification. In their research, Jonas Richiardi and Andrzej Drygajlo achieve a very high accuracy. In total 98.3% of all the signatures are classified correctly.
Dynamic time warping is discussed in [Martens and Claesen, 1996] and Mahalanobis decision making is used for classification. Ronny Martens and Luc Claesen, achieve in this way a correct classification of 98.4%.
Extreme points warping is a relatively new technique. In [Feng and Wah, 2003] this technique is developed and tested. Unfortunately it scores poorly on a signature dataset (74.6% correct classification). On the other hand, Hao Feng and Chan Choong Wah state that this technique is remarkably fast in completing the classification process, since it uses only the extreme points of a signature’s shape.
Hidden Markov models have been handled in for example [Dolfing, Aarts and Oosterhout, 1998], in which J.G.A. Dolfing, E.H.L. Aarts and J.J.G.M. van Oosterhout achieve a correct classification of between 98.1-99%.

Artificial neural networks are explained and applied on signatures in [Fuentes, Garcia-Salicetti and Dorizzi, 2002]. In their research, Marc Fuentes, Sonia Garcia-Salicetti and Bernadette Dorizzi use a fusion of hidden Markov models and artificial neural networks to reach a correct classification of approximately 91%. This is done via a Support Vector Machine (SVM).
Finally, in [Degerholm, 2005] different variants of artificial neural networks are applied to an Anoto pen. Unfortunately, it is not entirely clear how the features are obtained from the pen. Furthermore, the hardware that is used is not available anymore and it appears that expensive licenses are needed to link the digital pen to a Java translation servlet, which is able to convert the pen data to ASCII format. Although the results are promising (correct classifications between 94.5% and 98.5%), it is suggested to do more testing on signature verification techniques, before it can be applied in a real world environment.
The earlier mentioned five techniques will be discussed in more detail in chapter 0.

In this chapter, the term “correct classification” is used. Although this is one way of representing the accuracy of a technique, a more common used term is the EER, which stands for Equal Error Rate. The equal error rate is the rate at which there are an equal amount of false acceptances and false rejections. Consider, for example, that out of 100 signatures, 3 are falsely accepted and 3 are falsely rejected, the equal error rate is 3%. Note, however that false rejects and acceptances do not always have to be the same. In this case, the verification technique has to be fine tuned in order to produce equal values. This can be done by using a threshold. For example, if the false acceptance rate (FAR) is 9% and the false rejection rate (FRR) is 14%, the rule for accepting signatures has to be fine tuned in such a way that the false acceptance and rejections rates convert to an equal value. More information about the EER, FAR and FRR can be found in [BioID, 2004].

Note that the results mentioned in this chapter have been obtained under different experiment conditions. Unfortunately, research in which the same dataset is used for the different techniques mentioned earlier is not available. Despite the fact that the correct classification percentages mentioned do not tell much, they give an idea of how accurate a technique can be when the dataset is optimized. In section 6.7 the equal error rates for the mentioned techniques are summarized in a table, together with the dataset that is used.

5 Anoto technology

In this chapter an explanation of Anoto technology will be given. This is done to get a thorough understanding of the hardware that is used in this project and how this hardware could be linked to the signature verification techniques.

5.1 History

Anoto Group was founded as C Technologies in 1996 to realize the founder Christer Fåhræus' technological idea. Then in 1999, after several key mergers and partnerships had been formed, the Anoto Group was established. In 2000, Anoto Group AB was listed on the Stockholm Stock Exchange, where it still remains quoted on the Small Cap list of the OMX Nordic Exchange in Stockholm under the ticker ANOT.

Currently, the head office of Anoto Group AB is located in Lund (Sweden). Other offices are located in Boston (USA) and Tokyo (Japan). The group employs approximately 110 people, most of whom are located in Lund.[2]

5.2 Digital pen

One of the solutions in which Anoto Group AB has been successful is the digital pen and paper. The interested reader can refer to [AnotoHP, sd] for other solutions. For this project, the digital pen and paper are used, because handwritten signatures have to be captured using a pen and paper. Note that an Anoto digital pen does not have to mean that the pen is manufactured by Anoto Group AB. Digital pens have been manufactured by other companies, such as Logitech, Nokia and Maxell. These companies are all partners of Anoto Group AB and have a license to use Anoto functionality.

The digital pen is somewhat bigger than a normal ballpoint, but works exactly the same. A user can write on regular paper without noticing any difference compared to a ballpoint pen. The functionality of the pen, however, is not the same. To profit from Anoto functionality, the pen has to be used with Anoto grid paper (which is explained in the next section). The pen is able to recognize the paper and stores the coordinates of the current pen position, once a user starts writing on the grid paper. As we will see later, not only the coordinates but also other useful parameters are stored. For clarity, we depict the pen in Figure 5-1.

[2] Source: http://www.anoto.com/

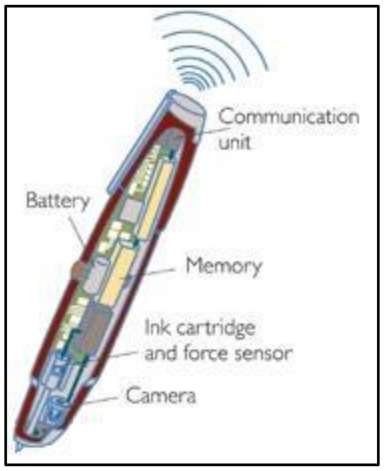


Figure 5-1: Digital pen with Anoto functionality

As we can see from Figure 5-1, the digital pen has an onboard memory unit. This unit is able to store up to approximately 50 fully written A4 pages. This amount will fluctuate depending on the digital pen manufacturer. In the bottom of the pen, a small camera is present that takes a certain amount of pictures per second continuously. From these pictures, coordinates can be extracted. Furthermore, an ink cartridge and a force sensor are installed. This force sensor is able to measure the pressure of the pen on the paper. This extra functionality is useful to gather information about the amount of pen strokes on the paper. The communication unit is able to send the stored data wirelessly to a device that contains the same wireless protocol. In this case, this protocol is Bluetooth. Not every manufacturer has installed this unit as a default. Next to the communication unit, the possibility to connect the digital pen to a computer by means of USB is present independent of the pen’s manufacturer. Finally, a battery is present, since the pen needs to operate wirelessly in order to feel close to a regular ballpoint.

5.3 Grid paper

As mentioned in the previous section, special paper is needed in order to store information in the digital pen. The paper is shown in Figure 5-3 below.

Figure 5-3: Anoto grid paper

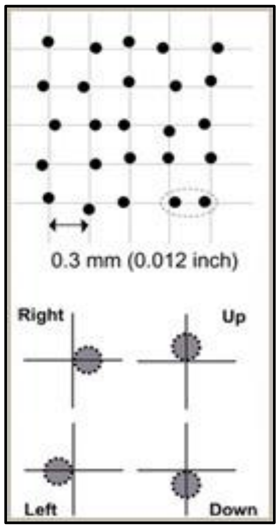


Figure 5-2: Anoto grid pattern

As we can see from Figure 5-3, the paper contains minuscule dots. These dots are printed in a pattern with a special printer that is able to print carbon ink under high resolution. The carbon ink is needed, because it absorbs infrared light. The camera in the pen is, in this way, able to identify the pattern, even through normal ink written or printed with non carbon ink. The pattern is shown in Figure 5-2 above. The space between the dots is approximately 0.3 mm and the dots are displaced according to the positions in the lower part of the figure. The digital pen uses only a small part of the grid (6 x 6 dots) to locate its position. Summing up all combinations of dot patterns it is possible to locate the pen's position if a user would write on a surface of 60 million km^2. This is equivalent to the surface of the continents Asia and Europe.

5.4 Anoto licenses

As mentioned before, Anoto technology covers only the capturing and storing of information from written material. Other digital pen and paper manufacturers can buy a license from Anoto and encapsulate the technology within hardware, in this case a digital pen and paper. Currently, there are over a hundred Anoto partners over the world, which use Anoto functionality and possess one or more licenses.

A license can be bought for the Anoto SDK, which contains a library with which the partner can create a custom application or issue commands and communicate with a digital pen. A second development kit is the form design kit (FDK), which is useful for defining forms. More licenses are available, but are not discussed here.

As an example, the company VisionObjects is taken, which is specialized in handwriting recognition. VisionObjects developed its own handwriting recognition engine (MyScript Builder) that is applied to different applications. One of these applications is called MyScript Notes. This application is able to load files produced by a digital pen and convert these files to digital characters. Furthermore, to improve the character recognition of a particular person, MyScript Notes includes MyScript Trainer. With MyScript Trainer a user can train the recognition software to improve handwriting recognition for that particular person.
To realize these applications, VisionObjects needed the Anoto SDK to read the files of the pen properly and extract information out of these files, since this is encrypted.

As can be seen, if an application has to be developed that relies on Anoto technology, a license to the SDK is needed. The interested reader can refer to [AnotoPP, sd] for detailed information about Anoto partners and the available licenses. The reference [VOHP, sd] contains information about the VisionObject organization.

5.5 Digital pen selection

In the previous section it became clear that, in order to access information from digital pens and create a customized application to access this information a license with Anoto Group AB is required. The ideal situation is to carry out this project with a minimum amount of costs.

As explained in earlier sections, different manufacturers of digital pens exist. In this section, a choice will be made which pen is used for signature verification. Currently, three Anoto partners regarding digital pens are present. These are Logitech, Maxell and Nokia. A fourth organization (Sony Ericsson) is also a manufacturer of a digital pen, based on Anoto technology, but is not listed as a partner by Anoto Group AB. Before making any statements about the digital pens, some of the properties of the different pens are gathered and shown in the table in Appendix B.

It can be seen that the four digital pens differ slightly from each other, but taking a closer look at the Sony Ericsson CHA-30, it can be noticed that it does not have a USB connection. For this project we need to connect a digital pen to the computer to load the data and run the signature verification. Bluetooth is an option, but a USB connection is much faster and compatible with most computers nowadays. This is the main reason why the CHA-30 is not suitable for this project.

The three remaining pens share many same features and in general any choice will suffice. However, the Maxell Penit is very difficult to obtain and delivery is likely to be postponed for weeks. Furthermore, it is known that the Logitech Io2 will work for this project, since the software can be downloaded that comes with the pen and some simple experiments with the software show that it captures written data in a correct way. Unfortunately, Logitech has stopped to produce digital pens. From this, using the Nokia SU-27W seems the most straightforward (and only) choice to make. This choice can be strengthen by the fact that the Nokia SU-27W uses the Logitech Io2 software, so the functionality is the same.

The decision to be made for the hardware used in this project is the Nokia SU-27W, since this is the ideal option, in terms of availability, compatibility and costs.

5.6 Software selection

Now that a decision is made which digital pen to use, software has to be selected that creates the possibility to connect the signature verification techniques to the digital pen.
In this section a brief description of the chosen software that is used is given. To conclude this section, a graphical representation of the hard- and software components used in this project will be presented.

The first application that is included with the digital pen is the Logitech Io2 software. This application allows reading data from the Nokia SU-27W. The software can be used for more purposes as can be seen in [Io2SW, sd], but will be used only for collecting signatures in this project.
MyScript Notes is a more complex tool that can learn writing styles of a users and convert written text to digital text. As with the Io2 software, MyScript Notes will be used for converting encrypted signature files to their decrypted (human readable) form. For the range of possibilities of MyScript Notes the interested reader can refer to [MSNotes, n.d.].

Next to the abovementioned software, additional software is needed to process the signatures. To implement signature verification techniques Matlab will be used, since this is a very powerful numerical computing environment. In addition, Matlab contains its own programming language and allows users to develop algorithms, visualize data, analyze data and execute intensive numeric computations. This has resulted in many user defined scripts that use the computing engine to realize certain functionality. These scripts can easily be customized to specific needs. Examples of scripts are: programs to compute prime numbers and dice rolling programs.
Another reason to use Matlab is that it offers a wide range of built-in methods for machine learning techniques. This means that powerful computations can be executed with a relatively small amount of instruction lines. Since the focus will lie in using these machine learning methods, Matlab is the ideal solution for signature verification. Matlab is not free and a license has to be obtained in order to use it. For more information about Matlab, see [Matlab, n.d.].

To prepare the signature data and use it as input for Matlab, noise has to be filtered out. C# together with ASP.NET has been chosen in this case. The main reason is the description of the project assignment, which states that the project should be realized using preferably Microsoft technology. A second reason is that, by using this technology, a user can upload the pen files all time (since a web server is deployed and the application is running on this server) and the files can be converted automatically to input for Matlab.
Visual Studio is required to implement the ASP.NET application. Visual Studio can be downloaded, installed and used freely for the first 90 days. After this period, a paid license is required.

In Figure 5-4, the combination of the software mentioned before is displayed.

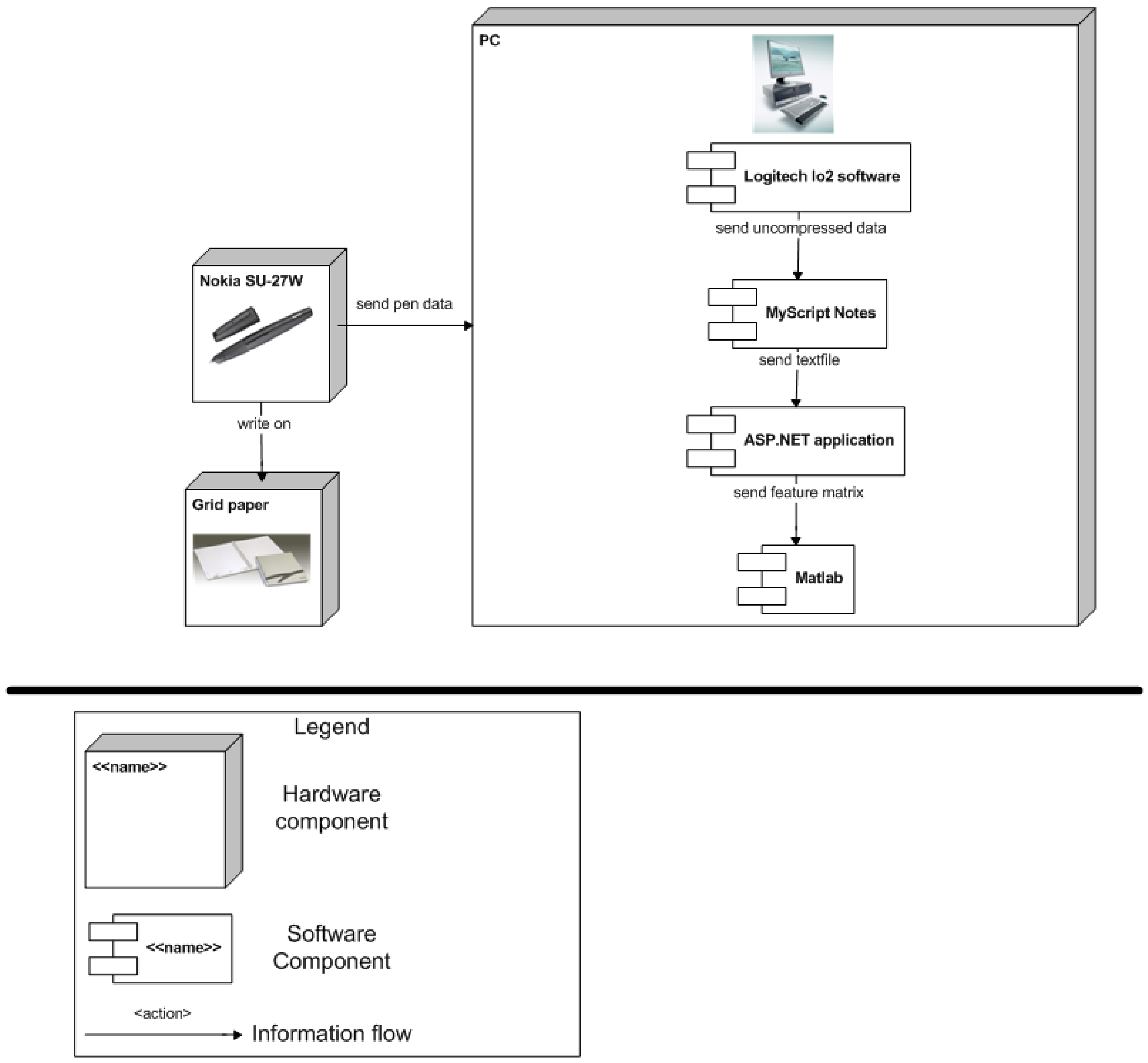


Figure 5-4: Hardware/Software relations schema

In the above figure, the hard- and software relations are displayed.
The Nokia SU-27W stores the information that has been written on a grid paper in its local memory unit and is able to send this information to a PC by means of a USB (or Bluetooth) connection. On the PC there are four applications that collaborate before applying a verification technique to the signature.

The Logitech Io2 and MyScript Notes application are needed in order to collaborate with each other and successfully convert the information to a format that is human readable. In this way the ASP.NET application knows which information to discard. Once discarded, the information that is left is put into a matrix, which on its turn can be exported easily to a Comma Separated Value (CSV) file. Matlab, on its turn can import CSV files and read the signature data successfully.
Note that MyScript Notes is not able to communicate with the Nokia SU-27W directly and the Logitech software is explicitly needed.

5.7 Digital pen information

As mentioned in earlier sections, the pen is able to store the drawn coordinates and measure pressure values. In this section the Nokia SU-27W will be used to draw a signature on grid paper. When this is done, the Logitech Io2 application is started and the pen is connected to the PC. Last, the data is transferred and converted with MyScript Notes to readable format files.

Each file contains several fields, which are straightforward to interpret, such as the x and y coordinates. These coordinates are stored in tens of millimeters.
Other fields may need a closer examination in explaining their function and are listed below.

Length of stroke
As explained earlier, the pen contains a force sensor which is able to measure the pressure values. In this way, separate strokes can be measured. The length of a stroke is defined as the number of x and y coordinates measured between a “pen down” and a “pen up” (i.e.: a stroke). A “pen down” is defined as the action of a user pressing the pen on the paper and a “pen up” is the action of the user lifting the pen from the paper. Unfortunately, no real-time pressure values are obtained. By this, it is meant that for every time unit a continuous pressure value that belongs to that particular time unit is captured and stored.
The only real-time values are the coordinates. The force sensor acts as a pressure switch, which is activated when the user touches the paper with the tip of the pen and deactivated when the user lifts the pen from the paper. In this way, a discrete value (“true” or “false”) is obtained referring to the pen touching or not touching the paper.
Real-time pressure values could have been useful when defining signature verification techniques, as is done in for example [Richiardi and Drygajlo, 2003].
The length of the stroke is put on a separate row, before the first x and y coordinate pair. In this way it is known how long the stroke is before reading the coordinates of that stroke.

Version number
On the top of the file a version number is displayed. This information gives an indication about the actuality of the hard- and software used, but is not useful in any way as input for the signature verification techniques.

Dimensions
Next to the version number, the file contains information about the width and height of the paper a user draws on with the pen. Consider, for example, the numbers 1248 and 876. These correspond roughly to a B7 format. Again, these numbers are in tens of millimeters. I.e.: the size is 12.48 x 8.76 cm.

Number of lifts
The file contains a number that represents of how many strokes the drawing consists. This can be useful, since a particular person may use a similar amount of strokes when signing a note.

Line color
The file can contain one or more (depending on the amount of strokes) decimals of length ten, followed by three decimals. The first is always of length ten, since it specifies the color value of the line. This has been determined during several tests with different colors in which the number changed when changing the line color.
Consider for example the decimal “4280877195”. When this decimal is converted to a hexadecimal the result is “FF29008B”. The first two characters have to be removed, since otherwise the color is not in a valid range. The presence of the first two characters guarantees a decimal length of always ten. For example, “29008B” corresponds to the decimal “2687115”, but another color “AAAAAA” corresponds to “11184810”. It can be concluded that the length differs and an inconsistency arises. When putting “FF” before a hexadecimal, the decimal length will always be ten.

Line thickness
The three numbers after the color value specify the line thickness. For example, this can be “3 3 0”. These numbers are displayed in hundreds of micrometers, so in this case the line thickness is 0.3 mm. It is unknown why the thickness is displayed two times and why the last number is zero. During different tests the last number always was zero and the first two numbers were equal. However, in not knowing these definitions, it poses no serious threats in the process of signature verification, since the line thickness is not a crucial feature.

Note that, despite the fact that the actual line thickness on paper differs within one signature, the line thickness in this context is purely software related. I.e.: The default is 0.3 mm, which means that no matter how thick the line on the paper is, the converted digital version has a constant line thickness of 0.3. This is why it will not affect the verification process.

End of file
The end of file consists of the last 5 rows. These rows are always the same. This can be very useful, since in this way the ASP.NET application can be implemented in such a way that it knows at which point a file ends when a user submits the file to the server.

This section will be concluded with an illustrative example to clarify which information is stored where in the digital pen.

```
v1.2
1248 876
3
1
4
4280877195 3 3 0
22
567 484
…
862 588
4280877195 3 3 0
41
316 557
…
387 555
4280877195 3 3 0
57
394 545
…
663 583
4280877195 3 3 0
44
441 522
…
114 600
0 0
0
70
0
0
```

Figure 5-5: pen data example

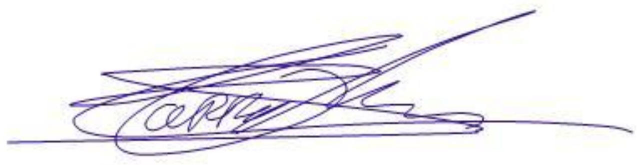


Figure 5-6: Corresponding signature

Looking at Figure 5-5, which contains the data that the digital pen stores after being converted to human-readable format with MyScript Notes, different lines are visible. For simplicity, only the first and last coordinate pair between two strokes (pen ups) have been displayed. For all intermediate coordinates the symbol "…" is used. The corresponding signature is shown in Figure 5-6.
On the first line in Figure 5-5, the version number is displayed (v1.2). The second line displays the height and width of the paper on which the signature has been made, in this case 1248 by 876. The third and fourth lines are still unknown and VisionObjects (the organization behind MyScript Notes) was unwilling to give a definition of these two integers.

The fifth line contains the integer number 4, which represents the number of strokes the signature contains. I.e.: the number of times a user lifted the pen from the paper when signing it. The sixth line contains the color of that stroke's ink. Note that this color is always physically the color of the ink the pen produces. For example, if the pen tip is blue, the color will always stay blue on paper, but we can tell the software to change the color to purple. In this way, the digital version (the one stored in the pen's memory) is purple and is displayed purple when transferred to a PC.

The three numbers after the color number are the line thickness numbers. Note that here the color number line also defines the beginning of a new stroke.
The seventh line contains the number of captured coordinates within that stroke, in this case 22. This can be a measure for how fast the stroke has been drawn compared to the other strokes (if there are any).
The eighth line contains the first coordinate pair of the first stroke ("567 484"). In this case after the eighth line, 20 lines with coordinates pairs follow and the 29th line contains the value "862 588", which is the last coordinate pair of the first stroke. The strokes that follow begin with the color line and are similar to the one described here.
The end of the file is characterized by the last 5 rows. Although it is not known why there are 5 rows and what they exactly mean, all files that have been used end with these 5 lines of which the content is identical.

5.8 Discussion

In this chapter, Anoto technology has been explored. The digital pen that has been chosen was analyzed and information that can be extracted from this pen became visible. Furthermore, the software that has to be connected to Anoto technology has been defined and configured. The signature verification techniques that are chosen rely on the choices that have been made in this chapter. Now that it is clear which hard- and software is connected to each other, signature verification techniques can be studied.

6 Analyzing signature verification techniques

Verifying signatures is a key task in this project and fortunately a lot of research has been done in this area. This has resulted in different techniques that each have their advantages and disadvantages. The challenge is to make a selection out of these techniques in order to proceed to the implementation of the selected techniques. In this chapter, different verification techniques that are eligible for selection will be analyzed.

6.1 Signature verification

In general, there are two types of signature verification approaches. The first type is offline (or static) signature verification. This type tries to verify signatures based on their static images. With this, it is meant that only the information about the image (e.g.: a bitmap image of the signature) is known. I.e.: It is only known what a user has signed (the result), not how he or she signed it.

On-line (or dynamic) signature verification, on the other hand, deals with the dynamic information of signatures. Examples are: the velocities at certain time stamps, the coordinate trace and pen pressure values. All these features are related to time so it can also be said that on-line signature verification is time dependant. A good overview in which both the on-line and offline case are considered can be found in [Plamondon and Srihari, 2000]. In their research, Réjean Plamondon and Sargur N. Srihari, propose different algorithms for on-line and offline handwriting recognition and make statements about the accuracy of both. One of their conclusions is that the on-line case is more accurate than the offline case. Note, however that handwriting recognition is observed, instead of signature verification. The basic idea however is the same, since the terms “on-line” and “offline” only refer to how much information can be extracted from writings.
The techniques that are studied in this project pertain to on-line signature verification. This choice has been made, since the results are more promising than with offline signature verification.

Earlier, it was stated that the results in Appendix A are not entirely fair, because of variations in the datasets and the methodologies used to test these techniques. For example, the size of signatures used varies a lot between the different suggested techniques and some techniques consider only random forgeries. These are forgeries that are made randomly without prior knowledge about the signature that is going to be forged. Although it appears that good results are yielded using some techniques, in this project more recent techniques are considered that have proven to yield similar results. These recent techniques consider skilled forgeries (the opposite of random forgeries), a larger signature database and feature selection that is close compatible with the hardware used in this project.

In this chapter, these more recent techniques will be explained in short in order to make a selection.

6.2 Gaussian Mixture Models

The Gaussian Mixture Models (or GMMs) technique is a statistical method that can be used for clustering low dimensional data with the help of several multidimensional Gaussian probability distributions. The construction of a GMM starts with a random initialization of cluster centers (means of Gaussian distributions) and their shapes (covariance matrices). The number of clusters (Gaussians) is a parameter that can be specified by the user. Next, a so-called Expectation Minimization procedure iteratively adjusts parameters of the mixture in order to maximize the correspondence between the data and the model. Once the model is trained, it can be used to measure the correspondence between sample data (test set) and the model. A more detailed description of GMMs and the EM procedure is given in later sections.

GMMs have been used intensively in the field of face authentication, such as in [Sanderson and Bengio, 2003a] and [Sanderson and Paliwal, 2003b]. Another popular field is speaker verification of which [Reynolds, Quatieri and Dunn, 2000] and [Stafford, 2005] are some examples.
Next to the above mentioned examples, GMMs are also popular within the field of signature verification, since good verification results can be achieved. In [Richiardi and Drygajlo, 2003], research has been done on a signature data set and an EER (Equal Error Rate) of 1.7 % can be achieved. For a more detailed explanation of GMMs the interested reader can refer to [Moore, 2005].

6.3 Dynamic Time Warping

Dynamic Time Warping, also known as DTW, can be used in signature verification to compare different signature to each other. According to [Martens and Claesen, 1996], DTW originates from the field of speech recognition and has been applied successfully in the field of signature verification more than once.

The problem with signatures is that most of them are of a different length. This makes comparison between local features hard if not impossible to realize. Dynamic time warping takes two signatures of different lengths as input and tries to find a matching between the points that are captured at certain time frames. All possible pairs are considered and an optimal time alignment between two signatures is calculated. This algorithm is more commonly known as an example of dynamic programming. In [Munich and Perona, 1999] a variant of DTW, called Continuous Dynamic Time Warping (CDTW) is developed and evaluated. However, this algorithm performs slightly worse than DTW. The EER of CDTW is 2.9 %, while that of DTW is 2.6%. In addition CDTW is three orders of magnitude slower than DTW.

6.4 Extreme points warping

Extreme points warping (EPW) is, similar to CDTW, a variant of DTW and developed in [Feng and Wah, 2003]. The strength of EPW is that it only tries to map the extreme points (the minima and maxima) of a signature's local features. In this way the computation time is decreased by a factor eleven. Furthermore, in [Feng and Wah, 2003] it is stated that EPW also reduces the EER, compared to DTW, but the EER's are rather high. The average EER of DTW and EPW are 33% and 25.4% respectively. These results, however, are based on the signatures' shape only. I.e.: only features that describe the shape of the signature are used for classification. This, in contrast with DTW, that is discussed in the previous section. It remains, however, unclear why the EER yielded in [Feng and Wah, 2003] is significantly higher than the EER in the previous section when considering only shape features (8.9 %). EPW is a promising signature verification technique due to its speed, but a lot still needs to be developed in order to improve the EERs.

6.5 Hidden Markov Models

A Hidden Markov Model (HMM) is, like a GMM, a statistical approach. An HMM takes the assumption, that the system to be modeled consists of Markov processes with unknown parameters. An HMM is seen as the most simple form of a Dynamic Bayesian Network (DBN).

The Hidden Markov Model (HMM) consists of the following elements:

- The number of states, N
- The number of observation symbols in the alphabet, M
- A set of state transition probabilities

The first element is the amount of samples taken from a signature. The second element represents the symbols that are used as states. With signature verification these symbols can be the x and y coordinates. The third element is the probability of transition from one state to another. The term "hidden" in hidden Markov models refers to that only the outcome (not the state) is visible to an external observer. In the next figure, a simple HMM is shown.

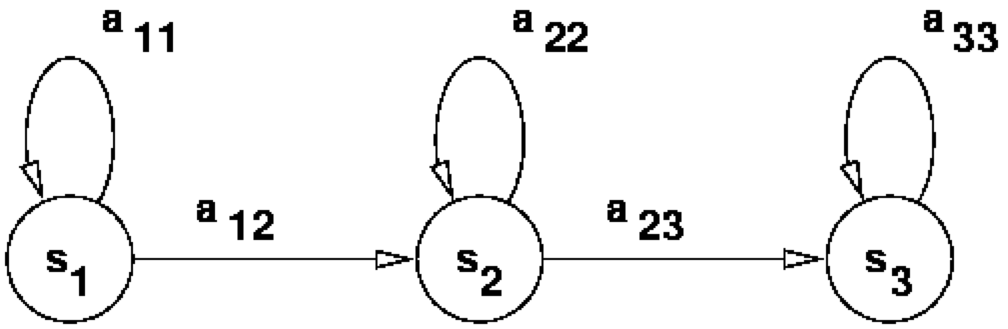


Figure 6-1: HMM graph

In this figure, there are three states and five transition probabilities. The state transition probabilities a_{ij} can be calculated by using the following formula:

$$a_{ij} = p\{s_{t+1} = j | s_t = i\}, 1 \leq i, j \leq N$$
$$With:$$
$$s_t, the\ current\ state$$
$$a_{ij}, the\ transition\ probability\ between\ state\ i\ and\ j$$

Although the specification of an HMM may seem easy to understand, when exploring this technique in more depth certain problems arise. The basic problems of HMMs are the evaluation, decoding and learning problem for which algorithms have been suggested to approach these problems. Since the purpose of this section is to give a general understanding of what HMMs are, these problems will not be explored in depth. The interested reader can refer to [Rabiner, 1990] and [Warakagoda, 1996].

In [Lv and Wang, 2006], signature verification is done analogously to [Richiardi and Drygajlo, 2003] with the difference that for the verification task HMMs are used. It is possible to obtain an EER of 1.5 % chaining HMMs together. In [Yang, Widjaja and Prasad, 1995] an application of HMMs is proposed that yields a false rejection rate (FRR) of 1.75 % and a false acceptance rate (FAR) of 4.44 %. The EER will lie somewhere in between these values. These results are promising, considering the year in which the research has been performed. It should be noted however that only simple type forgeries are considered. In [Kashi, et al., 1998] a research is being done regarding signature verification using an HMM. In this research project Fourier normalization is used on the first difference of the coordinates that describe the signature's shape. In this way, an EER of 2.5% can be obtained. Finally, in [Dolfing, Aarts and Oosterhout, 1998] another HMM approach is taken that yields an EER between 1.0% and 1.9%.

6.6 Artificial Neural Networks

Artificial Neural Networks (ANNs) are inspired by how the human brain learns. ANNs consist of neurons, which are units that take several observed inputs and produce a single output. This is shown in the figure below.

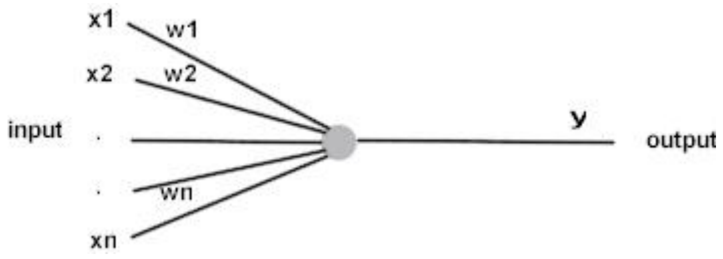


Figure 6-2: Single-layer perceptron

The collection of inputs, weights, neuron and output is reffered to as a perceptron. In this case a single layer perceptron is defined, since only one neuron is present. The neuron takes an input and multiplies it with a weight. The sum of all inputs times their weight is taken minus a fixed threshold. This can be specified as follows:

$$output = \sum_{i=1}^{n} w_i x_i - \theta \tag{1}$$

$With:$

$w_i, the\ weight\ corresonding\ to\ input\ i$

$x_i, the\ input$

$\theta, the\ fixed\ threshold$

The single-layer perceptron is the simplest form of an ANN and can be used to define a multilayer perceptron, which is more sophisticated as is shown below.

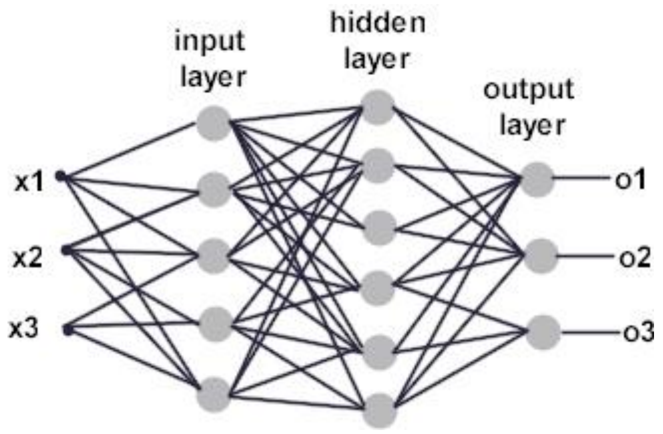


Figure 6-3: Multilayer perceptron

As can be seen, a certain number of perceptrons are organized into layers: one input layer (the leftmost one), one or more hidden layers and one output layer (the rightmost one). The input values $x_1 \dots x_n$ are propagated forward from left to right, producing a vector of output values $o_1 \dots o_m$ (the number of inputs does not have to be equal to the number of outputs).
The hidden layer can contain one or more hidden neurons that enable the network to learn complex tasks by extracting progressively more meaningful features from the input patterns[3].
The network is trained by propagating the input patterns forward to calculate the output (forward propagation), the error (the difference between the desired and actual output) and the propagation of this error backward (back propagation) to adjust the weights.
Finally, classification can be realized by defining a rule on the output. For example, if the output is +1, the point represented by all the inputs is assigned to class 1. On the other hand, if the output is -1, it will be assigned to class 2.
A lot more can be discussed on artificial neural networks. Here, the general idea is provided. More information can be found in [Haykin, 1994] and [Stergiou and Siganos, 1996].

In [Quek and Zhou, 2002] a minimum EER of 8.96 % is achieved by choosing an optimal set of features. Note that this EER is achieved in the field of offline signature verification in contrast with online signature verification, which is used in this project. In [Bajaj and Chaudhury, 1997] an EER of 3% is achieved using offline signature verification as well, 150 signatures and 100 random forgeries. This in contrast to the former result of 8.96, which includes 535 signatures from different ethnic groups and more than 60 skilled forgeries. This explains the higher EER. Finally, in [Fuentes, Garcia-Salicetti and Dorizzi, 2002] an online signature verification approach is taken using both an HMM (explained earlier) and a neural network. This yields an EER of 5.87% and takes random as well as skilled forgeries into account.

[3] [Haykin, 1994]

6.7 Discussion

In this chapter, different signature verification techniques have been analyzed and explained shortly to create an overview of existing research.

The techniques that are examined yield different results. It should also be noted, however, that the EERs mentioned depend a lot on the data set used for training and verification. For example, the amount of sample and test signatures used and the type of signatures. This does not mean that the EERs are completely useless. They should be a good approximation for the results expected in this project. All in all, the EERs should be considered as a guideline for the results of this project.
To conclude this chapter, a table is given in which the EERs are displayed of the earlier mentioned techniques. In addition, the amount of genuine signatures and the number of skilled and random forgeries are displayed. These statistics are displayed in Table II. It should be emphasized that only dynamic data is considered.

	Equal Error Rate (EER) in %	Number of signatures	Number of Skilled/Random forgeries
Gaussian Mixture Model (GMM)	1.7	1000	1250 skilled forgeries
Dynamic Time Warping (DTW)	1.6	360	Unknown amount of only random forgeries
Extreme points warping (EPW)	25.4	750	250 skilled forgeries
Hidden Markov Model (HMM)	1.0-1.9	1530	3240 skilled forgeries[4]
Artificial Neural Network (ANN)	5.87	1530	3200 skilled forgeries[4]

Table II: Summary of EERs

[4] Subdivided into three different skill levels

7 Selecting signature verification techniques

In this chapter a selection of the previously mentioned signature verification techniques is made. This selection consists of two techniques of which I believe (based on the results in Table II) they perform well in the Anoto environment. The reason only two techniques are chosen is mainly due to the time that is available for this project. Another reason is that a lot of research already has been done as can be seen from the amount of techniques that are available, shown in Appendix A. Furthermore, using all these different techniques in the Anoto environment is meaningless, since techniques which perform significantly worse can be left out. Again, the purpose in this research is to yield a low EER (Equal Error Rate), instead of comparing different signature verification techniques. The interested reader can, however, refer to [Plamondon and Lorette, 1989], which contains additional comparison tables.

7.1 Selection of techniques

Looking back at the different techniques, two groups can be defined. One group contains the statistical techniques (hidden Markov models, Gaussian mixture models and artificial neural networks) and the other group contains a direct mapping technique (dynamic time warping and extreme points warping). It can be argued that neural networks lie more in between the two groups instead of pertaining purely to the statistical techniques. Note that the term “direct mapping” is defined here to indicate the method of mapping signatures of different lengths to each other.
Preferably, one technique of each group should be chosen in order to evaluate different approaches.

According to the results in the previous chapter, Hidden Markov Models (HMMs) and Dynamic Time Warping (DTW) should be used. However, the difference in accuracy, between Gaussian Mixture Models (GMMs) and HMM is minimal as explained in [Richiardi and Drygajlo, 2003]. Here, GMMs are chosen as a primary technique, because they can be linked to the Anoto environment fairly easy.
Artificial neural networks are not part of the research in this project, since they yield a higher EER and the coupling to Anoto environment is more complex.
In the other group, neither DTW (Dynamic Time Warping) nor EPW (Extreme points warping) is going to be used. Instead a variation is used, as defined in [Gupta and Joyce, 2007]. The downside is that it achieves at minimum an EER of 2.5%, which is higher than DTW (but lower than EPW). This creates, however, the opportunity to improve the technique and try to achieve a lower EER, since this technique is fairly new. Another advantage is that this technique has the speed of EPW, resulting in a much shorter computation time compared to DTW. The eventual goal is to find a faster and more accurate classification method than DTW. In this case, a step in that direction is done by using and improving the technique proposed in [Gupta and Joyce, 2007].

In the following sections, Gaussian mixture models and the extreme point warping variant will be deepened out.

7.2 Extreme points warping Variant

Having briefly highlighted DTW (Dynamic Time Warping) in section 6.3 and EPW (Extreme Points Warping) in section 0 and knowing that the Extreme points warping variant, from now on called EPWV, is inspired on these two techniques, EPWV is the next technique that will be highlighted. EPWV is described in [Gupta and Joyce, 2007] and this source will be used mainly to analyze and clarify the technique in the next sections.

7.2.1 Feature selection

EPWV assumes the following feature data to be handled as input:

- X coordinates
- Y coordinates
- Pressure values

The main problem is that with the current hardware that is used, the pressure values are not obtainable, since the pen does not keep track of these values. Instead, another approach has to be found in order to maintain and improve the EER of EPWV.

7.2.2 Training the model

Before actually training the model, the signature data is first analyzed and converted to a string representation.
The first step is to scan the x coordinates for local minima and maxima. This step is also done for the y coordinates. The points on which the pen is lifted from the paper are also recorded. This information is then converted to a string. Keeping only the minima and maxima is the reason for the decreased computational time of the technique. Discarding, in this way, all other coordinates prevents the algorithm to scan for each x and y coordinate combination as is done in the traditional dynamic time warping algorithm in [Martens and Claesen, 1997]. In this algorithm two “for” loops are defined in order to find the optimal path between the coordinates of two signatures. The EPWV technique also contains two “for” loops. However, the main difference is that the string containing the minima and maxima has much fewer points (at specific time stamps) than the original signature data. This will result in a faster completion of the algorithm. Note that not the coordinates are mapped to each other, but the strings of different signatures. Hence, another algorithm is needed to map the strings to each other and find an optimal match. This algorithm is described in [Wagner and Fischer, 1974]. The basic idea is described in [Gupta and Joyce, 2007] and will be explained next for clarity.

To verify an input signature, its string representation is compared to the string representations of each signature in the database. Unfortunately, the strings are rarely of equal length, which means an algorithm is needed that compares the difference between strings of different length. Fortunately, there exists such an algorithm and is known as the Wagner Fischer algorithm ([Wagner and Fischer, 1974]). In this project, it is chosen for a C# implementation, for easy integration with the ASP.NET application that converts the signature coordinates to a string representation.

The distance between two arbitrary strings can be defined as follows:

$$d_{ij} = d\big(A(1,i), B(1,j)\big) \tag{2}$$

$$\begin{aligned}&With:\\ &i\ and\ j\ the\ i^{th} and\ j^{th} position\ in\ the\ string\end{aligned}$$

$A(1,i)$ is specified to denote a row vector or, more commonly, an array of characters.
This will be explained considering the next two strings:

- Foo
- Four

According to [Wagner and Fischer, 1974], the resulting diagram can be defined as follows:

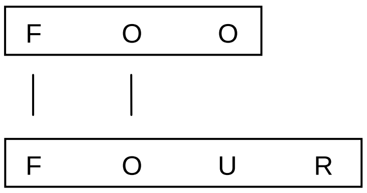


Figure 7-1: String to string comparison example

Now, let's assume that the cost for insertion and deletion of a character is defined to be 1. In addition, the cost for substitution is 1. A matching character has a cost of 0, since the distance from a character to an identical one should not incur any costs.

As can be seen from the figure above, the first two characters are identical. I.e.: No costs are being made. However, when the algorithm reaches the third characters of the first and second word, it can make two decisions. The first decision is to delete the "O" and insert the "U". According to the cost definition given earlier this adds up to 2. The second decision is to replace the "O" by the "U", resulting in a less cost of 1. In this case, substitution is chosen. Note that the decisions depend on the cost definition. If the cost for substitution is changed from 1 to 2, then the first decision results in the same cost as the second decision.

When the algorithm reaches the "R" and sees the first word has ended it can conclude that the insertion of an "R" at the end of the first word results in the second word. This equals a cost of 1. The total cost is 2 and this value is returned as the result of comparing these two strings.

The algorithm

The algorithm that is used for the comparison is based on [Mistrut, 2002], in which Dree Mistrut gives an implementation in Perl. The baseline of the algorithm is the definition of the cost matrix. In this matrix, the costs for the possible transitions are made. Following the example defined earlier, the cost array can be defined as [0, 1, 1], referring to the costs of a matching character, deletion/insertion and substitution respectively.
The algorithm starts with filling the first row and column with the costs of insertion/deletion times the index of the row or column as shown below:

$$cost\ matrix = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 1 & & & \\ 2 & & & \\ 3 & & & \\ 4 & & & \end{bmatrix}$$

During initialization of the cost matrix, the amount of columns is equal to the length of the first word plus one. Analogously, the rows of the matrix equal the length of the second word plus one. Once the initialization is complete, the algorithm starts looking at the first character of the first and second word.
The next step is to determine the costs of the remaining characters and calculate a minimum distance path. The costs of a position in the cost matrix are defined by the following equation:

$$d_{ij} = \min\{d_{i-1,j} + del(A_i), d_{i,j-1} + ins(B_j), d_{ij} + sub(A_i\ with\ B_j)\} \quad (3)$$

Roughly, what the algorithm does, is to determine the minimum cost to take at a certain point in the matrix. It knows where the deletions, insertions and substitutions are and compares these with the corresponding costs. Eventually, at position i,i (the lower right corner of the cost matrix) the cost for the minimum distance is displayed.

Following the example of the two strings "foo" and "four" the first outcomes of equation (3), in which I and j are both 1, are the following:

1. $d_{i-1,j} + del(A_i) = 1 + 1 = 2$
2. $d_{i,j-1} + ins(B_j) = 1 + 1 = 2$
3. $d_{ij} + sub(A_i\ with\ B_j) = 0 + 0 = 0$

According to these outcomes, option 3 is chosen, since this is the minimum value and the cost matrix is updated. The result is shown below:

$$cost\ matrix = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 1 & 0 & & \\ 2 & & & \\ 3 & & & \\ 4 & & & \end{bmatrix}$$

Eventually, near completion, the algorithm will produce the following cost matrix, in which the outcome is shown at the bottom right corner.

$$cost\ matrix = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 1 & 2 \\ 2 & 1 & 0 & 1 \\ 3 & 2 & 1 & 1 \\ 4 & 3 & 2 & \mathbf{2} \end{bmatrix}$$

The interested reader can refer to Appendix C for the C# code of the algorithm that returns the minimum edit distance.

Now that the Wagner Fischer algorithm has been defined, the following steps are defined (according to [Gupta and Joyce, 2007]) in order to verify a given input string.

1. Select five sample signatures (out of fifteen) randomly per user and find their string representation. These are the genuine signatures that are present in the database.
2. Out of these five samples, create ten pairings of two signatures each and compute the distance. After having obtained ten distances, calculate the mean and standard deviations of these ten distances.
3. For the signature that has to be tested, find the string representation.
4. Compare the test signature with each of the five sample signatures and compute the mean distance.
5. Compare this mean with the mean vector that is obtained in step 2 plus a threshold times the standard deviation vector. Accept the signature if the result of step 4 is smaller than the computed result in this step (with the threshold). Reject otherwise.
6. Repeat steps 4-6 for the genuine signature set (in this way the technique is tested if it accepts the genuine signatures) and the forgery attempts (in this way the technique is tested if it rejects the forgeries successfully).

7.2.3 Testing the model

To test the model for correctness, the steps 4-6 described in the previous section have to be carried out. Once this is done, the EER can be determined. As described in [Gupta and Joyce, 2007], different tests have been executed. Initially, the EER was at worse 14%. Some improvements were made resulting in a minimum error rate of eventually 2.5%. These improvements are the following:

1. The addition of magnitudes.
2. Adding global features.
3. Length of time gaps between extreme points.
4. Using individual thresholds.

The first improvement adds an extra character if a minimum or maximum is high or low compared to another minimum or maximum. For example, if a maximum is two times higher than another maximum encountered earlier, this higher maximum contains two characters in order to define it to be relatively high.
The second improvement introduces a combination of some global features, such as total time, number of sign changes in the x and y velocities and accelerations, pen-up time and total path length. Unfortunately, due to the limitations of the pen, the pen-up time cannot be measured. The other global variables can be measured and used for further research.
A third improvement is the introduction of time gaps between extreme points. In this way, it can be told how large the time gap between two consecutive extreme points is. This addition of information further improves the EER.
The fourth improvement suggests using individual thresholds instead of one global threshold value. If this threshold is chosen well, an EER of 2.5% is reached.

7.2.4 Discussion

EPWV was discussed and it shows some promising features. EPWV is an easy to understand, yet accurate technique to use in signature verification. The technique is relatively new and preferably has to be improved to achieve a lower EER. This will make it a serious competitor among other, more mature techniques, such as GMMs. Unfortunately, not all data can be captured by the pen and this flaw will become visible during the experiment as this can have an impact on the EER. The challenge is to obtain an equal or better EER with the data that is obtainable by the digital pen.

7.3 Gaussian Mixture Models

As explained earlier, a GMM (Gaussian Mixture Model) is a statistical classification approach that can be used in the field of signature verification. Below, the different phases of the signature verification task are defined.

7.3.1 Feature selection

In [Richiardi and Drygajlo, 2003] it is suggested to select the following local features:

- X coordinates
- Y coordinates
- Pressure values
- Local velocities
- Trajectory tangent angles

As mentioned in the EPWV section, unfortunately the pen is not able to obtain real-time pressure information. The x and y coordinates, however are obtainable by reading the pen data into a computer. The result is one line of x and y coordinates per snapshot taken.
Local velocities have to be calculated from the amount of snapshots and coordinate values that are obtained. Velocities can be obtained by comparing the displacements of coordinates within one signature to each other.

To clarify this process the following formula is defined:

$$v_t = \frac{\sqrt{|\Delta x(t)|^2 + |\Delta y(t)|^2}}{\Delta t} = \frac{\sqrt{\left(x(t+\Delta t) - x(t)\right)^2 + \left(y(t+\Delta t) - y(t)\right)^2}}{\Delta t} \quad (4)$$

$$\begin{aligned} &With: \\ &x(t+\Delta t), the\ x\ coordinate\ at\ time\ t+\Delta t \\ &x(t), the\ x\ coordinate\ at\ time\ t \\ &y(t+\Delta t), the\ y\ coordinate\ at\ time\ t+\Delta t \\ &y(t), the\ y\ coordinate\ at\ time\ t \\ &\Delta t, the\ time\ period\ in\ which\ the\ measurement\ has\ been\ taken \end{aligned}$$

This formula calculates the Euclidean distance of the coordinates and does this for every time instance until the end of the signature has been reached. In this way a velocity is obtained that represents the general displacement in the x and y axis. Note that this is done according to the research described in [Richiardi and Drygajlo, 2003] and is proven to yield a low EER (Equal Error Rate) eventually. It can be discussed if splitting up the general displacement velocity in two separate x and y velocities yield better results.

Finally, the trajectory angles also have to be calculated from the coordinates. The formula is specified as follows:

$$\theta_t = \tan^{-1}\left(\frac{\Delta y(t)}{\Delta x(t)}\right) = \tan^{-1}\left(\frac{y(t+\Delta t) - y(t)}{x(t+\Delta t) - x(t)}\right) \quad (5)$$

This feature has identical symbols to equation (4) and can be very useful to determine the curvature of a certain segment within the signature. For example, in the case of two signatures with different heights and widths that belong to the same user. The curvature could prove to make the difference in accepting the signature, since it will not change much compared to the coordinates. I.e.: The similarity of the two signatures is found in the trajectory angles feature.

In the experiment of this project only x and y coordinates are used as features for input data. The main reason for this restriction was time. The theory behind GMMs in this chapter, however, will use a dataset of x and y coordinates, pen velocity and trajectory angles. I.e.: the input data is (x, y, v, θ) as can be seen in the next section.

7.3.2 Training the model

To train the model, an input vector is needed, which is defined as follows (according to [Richiardi and Drygajlo, 2003] and [Lv and Wang, 2006]):

$$M = \begin{bmatrix} x_1 & \cdots & x_t \\ y_1 & \cdots & y_t \\ v_1 & \cdots & v_t \\ \theta_1 & \cdots & \theta_t \end{bmatrix} \qquad (6)$$

The input vector is actually a matrix with 4 rows and t columns. As mentioned before, the pressure vector cannot be obtained or calculated. The values in the matrix have to be normalized to a zero-mean unit variance distribution. This is done, because the ranges of the features in the matrix are expected to differ a lot. By normalizing the data, a new matrix is obtained that describes how much each element strays from the mean of the feature to which that element belongs, defining in this way a cluster. According to [Richiardi and Drygajlo, 2003] and [Lv and Wang, 2006], this normalization is done as follows:

$$\mathrm{N}_M = \frac{M_{ij} - \mu_{M_i}}{\sigma_{M_i}} \qquad (7)$$

$With$:
μ_{M_i}, $the\ mean\ of\ a\ particlar\ feature$
σ_{M_i}, $the\ standard\ deviation\ of\ a\ particular\ feature$

After the normalization has been done, the matrix N_M can be used as input for the GMM. The curious reader can refer to [Richiardi and Drygajlo, 2003], [Lv and Wang, 2006] and [Koh Chin Wei, 2006] for more information about the Gaussian Mixture Model. Here, the details are left out, since the purpose is not to define a GMM, but instead to understand the basic idea and use it for signature verification.
The challenge with GMMs is to initialize a model using the right number of mixture components, which define the centers of data clusters. In [Richiardi and Drygajlo, 2003] these numbers are found to be 32 and 64. In this project it is not yet known how much mixture components will yield a feasible result, but these numbers are taken into consideration.
After the number of components have been determined, the model means are estimated using the K-means algorithm. This estimation is done randomly to achieve a fast convergence.

Once the GMM has been initialized, it needs to be trained using the Expectation-Maximization (EM) algorithm, which re-estimates the component weights, means and variances. This is done to find the maximum likelihood estimates of the means, covariances and the mixing coefficients (or prior probabilities of the means) of the Gaussians in the model. Once this has been done, the model is ready to be compared to other models.

7.3.3 Testing the model

To test the model on a given signature, the proposed method in [Richiardi and Drygajlo, 2003] is used. The following formula is used:

$$S(O,\Theta^{\mathrm{u}},\Theta^{-}) = \log \mathrm{p}(\mathrm{O}|\Theta^{\mathrm{u}}) - \log \mathrm{p}(\mathrm{O}|\,\Theta^{\mathrm{World}}) \tag{8}$$

$With$:
$O, the\ observation\ of\ a\ given\ signature$
$\Theta^{\mathrm{u}}, the\ GMM\ of\ a\ specific\ user$
$\Theta^{-}\ and\ \Theta^{\mathrm{World}}, one\ GMM\ created\ from\ the\ individual\ GMMs\ minus\ \Theta^{u}$

I.e.: in order to compute the signature score (how well an input signature resembles a particular signature stored in the signatures database) several steps have to be taken.
First, the probability that the input signature data belongs to a specific user model is calculated.
Second, the probability that the input signature data belongs to one of the other user models is calculated. The log functions are subtracted from each other to obtain the ratio between the first and second step, since $\log A - \log B = \log \frac{A}{B}$.

7.3.4 Discussion

In the previous sections, Gaussian mixture models were highlighted and the different steps of this technique became visible. As described earlier, the approach taken in [Richiardi and Drygajlo, 2003] is used in order to test the technique on signatures. Most of the formulas in the previous sections are derived from [Richiardi and Drygajlo, 2003] and [Lv and Wang, 2006], with some minor differences. They are redisplayed and explained again to clarify the role these formulas play in this project. Not all information that is handled in the above mentioned references is displayed here, since the main purpose is (as explained earlier) to clarify the crucial parts of GMMs in general and not to dive into the depths of the technique.

8 The experiment

Now that it is known which techniques are selected and how these techniques work in theory, a coupling to Anoto environment has to be made. In addition, to realize the techniques, new algorithms are needed and have to be developed. In this chapter, a description of how the techniques are applied to Anoto environment is given.

8.1 Data collection

Before starting to apply the two signature verification techniques that are chosen for the experiment, signatures are collected from different persons. 18 persons were asked to produce 5 genuine signatures each, producing a total of 90 genuine signatures. Furthermore, every person was asked to study carefully the signature samples of another person and mimic the signature 2 times, resulting in a total forgery set of 36. Although this set is small in comparison to signature sets, such as in [Richiardi and Drygajlo, 2003], a good impression can still be obtained about the accuracy of the techniques in Anoto environment. In the figure below, the process of collecting data is shown.

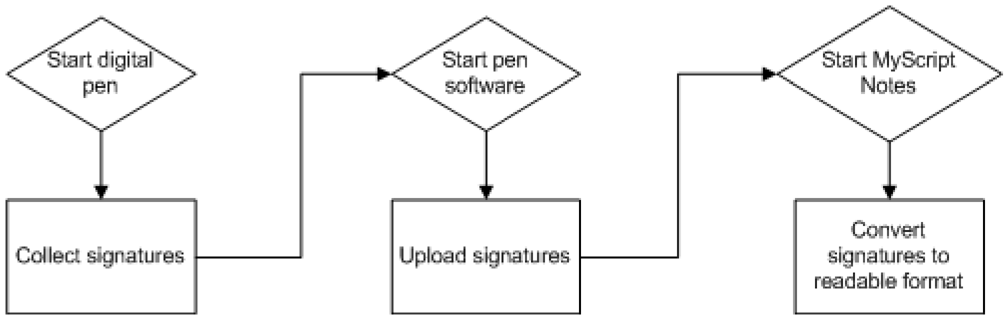


Figure 8-1: Collecting signature data

In the figure, the diamond shapes represent the execution of different software applications that are used. The rectangular shapes represent the tasks for which the software applications are used. The arrows show in which order the tasks are executed.
The process starts with the digital pen and the collection of signatures, which is described above. Once all signatures have been collected, the pen is connected to a desktop computer in order to upload the signatures with the help of the Logitech Io2 software. These signatures are encrypted and not readable. Using MyScript Notes, the signatures can be decrypted to a readable format. Unfortunately this can only be done manually, since the application is not open source and cannot be customized in order to automatically convert all uploaded signatures into a readable format. This means that 90 genuine signature files have to be opened into the application and saved to a readable format. This is a cumbersome procedure that takes some time, since a total signature set (including the forgery signature files) of size 126 is present. Especially when, in the future, new signatures are added the dataset becomes very large and the time to convert the signatures manually increases significantly. A good solution would be to

automate this process and create a method that converts the signatures in batch one by one, using MyScript Notes.

Although converting the signature data requires manual operations that may take a large time to complete, once this conversion is done, the preparation of data and subsequent phases of the verification can be done automatically as is explained in the next sections.

8.2 Data preprocessing

In section 5.7, the description of the available pen data was given. From this description, it can be concluded that the only available dynamic information (information related to time stamps), are the x and y coordinates. From this information, the speed and trajectory tangent angles can be derived. As explained earlier, in this experiment only the x and y coordinates are used for classification, due to the available time. Additional features have been left out for the moment and can be added in a future approach.

When Gaussian mixture models and the extreme points warping variant have to be applied, not all information from the digital pen is needed. Therefore, irrelevant information has to be filtered out. The filtering of information depends on which technique is used. The EPWV technique, for example, can additionally use the pen lift information about a signature to improve accuracy. The difference of signature information between the two techniques will become clear in subsequent sections. For now, assume the two techniques will not have exactly the same features, but will differ to some extent from each other.
The process of selecting certain features (and filtering out irrelevant information) is more commonly known as data preprocessing. There exist three different feature selection methods as defined in [Richiardi, Ketabdar and Drygajlo, 2005] and [Rhee, Cho and Kim, 2001]. These will be described next.

8.2.1 Global feature selection

With global feature selection one feature is extracted for the whole signature. This means that the x and y values have to be shrunk to one value. This can be, for example, the mean of the x and y values. In Figure 8-2 below an example of global feature selection is shown.

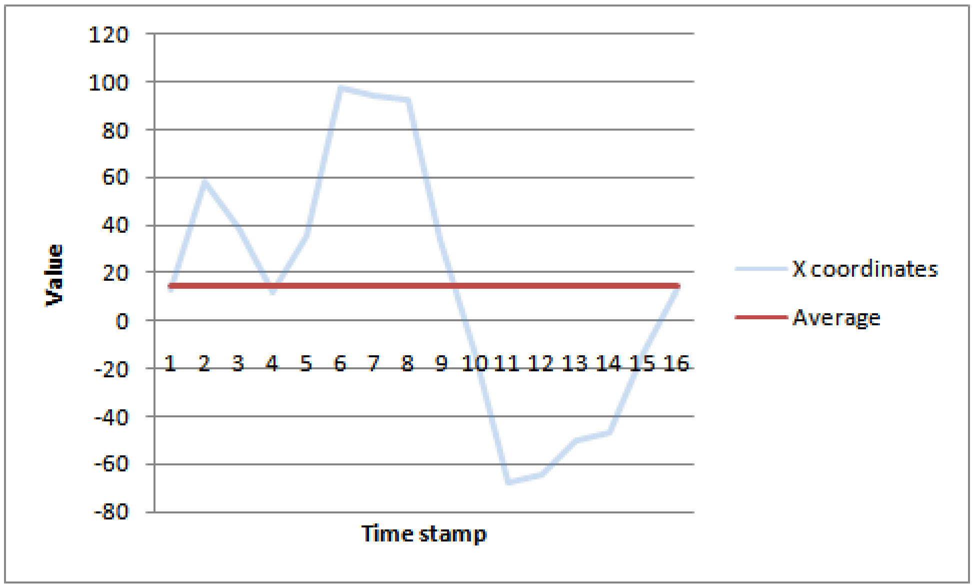


Figure 8-2: Global feature selection

As can be seen from the figure, the average of all the x values is taken. This results in the number 14.8125 and is added to the feature vector. In this way, one final vector is obtained that contains different global features of the signature. Intuitively, this leads to a decreased performance, since information is being lost. However, fairly good results can still be obtained and the execution time is less than using local feature selection. Local feature selection, on the other hand, still outperforms global feature selection when looking at the EER (Equal Error Rate) of the classification techniques. Research regarding global feature selection is given in [Fierrez-Aguilar, et al., 2005], [Ketabdar, Richiardi and Drygajlo, 2005], [Lee, Berger and Aviczer, 1996] and [Richiardi, Ketabdar and Drygajlo, 2005].

8.2.2 Local feature selection

With local feature selection, the input data remains intact. By this, it is meant that all x and y coordinates are used as input for the signature verification technique. This also means that all other features, such as speed, have to be of the same length. If the length between features differs, an inconsistency arises and the verification technique cannot be considered reliable anymore. I.e.: the speed has to be calculated for each time instance, corresponding to the time instances of the coordinates. In Figure 8-3 below a similar example of that in the previous section is shown.

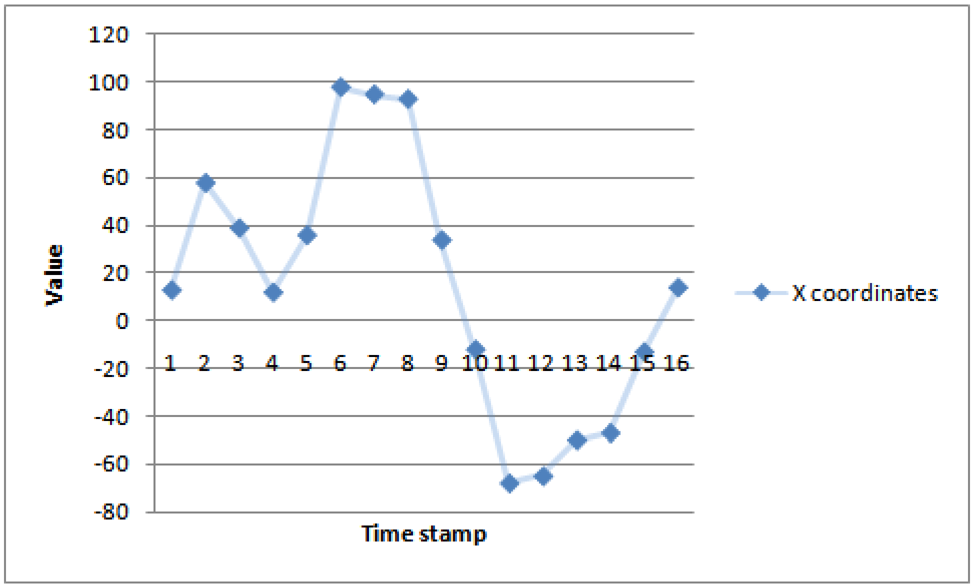


Figure 8-3: Local feature selection

In the above figure, 16 coordinates are shown that represent X values of the signature. Local feature selection keeps all these 16 points and puts them into one feature vector, say X, resulting in:

$$X = [13 \quad 58 \quad 39 \quad 12 \quad 36 \quad 98 \quad 95 \quad 93 \quad 34 \quad -12 \quad -68 \quad -65 \quad -50 \quad -47 \quad -13 \quad 14]$$

In contrast with the previous selection technique, a matrix will be obtained with a row for every feature and a column for every time instance as suggested in [Lv and Wang, 2006] and [Richiardi and Drygajlo, 2003]. [Guo, Doermann and Rosenfeld, 1997], [Kholmatov, 2003] and [Kholmatov and Yanikoglu, 2005] describe in more detail the process of selecting local features.

8.2.3 Segmental feature selection

This feature selection method is a combination of the previous two. It divides the signature into different segments and one feature is extracted for every segment ([Richiardi, Ketabdar and Drygajlo, 2005]). This will result in a matrix as in the local feature selection method. The main difference is that the matrix will be smaller, assuming one segment contains more than one feature (i.e.: more than one time instance). This can be best explained with an example illustration. The illustration is shown in Figure 8-4 below.

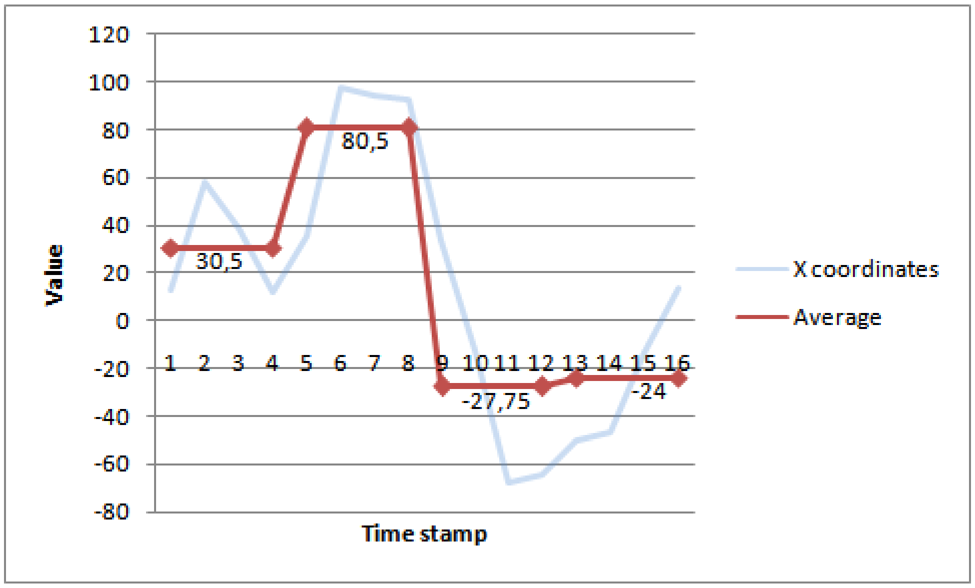


Figure 8-4: Segmental feature selection

In the above displayed figure (which lists sixteen captured x coordinates), four segments are defined. Each segment contains the average value of four x coordinates that belong to that segment. This results in four different segments that each has one average value as is displayed in the red graph. For example, the first segment “30,5” is the average value of the first four x coordinates in the blue graph.
In this way, instead of having sixteen features (as with local feature selection), only the four features from the red graph are stored, resulting in a smaller matrix. On the other hand it contains more information (four features) about the signature compared to the global approach (one feature).

Depending on the selection of segments this method will provide a less complicated approach than the local method, while being more discriminating and robust than the global method. The scientific theses [Dolfing, Aarts and Oosterhout, 1998], [Lin and Chen, 1998] and [Rhee, Cho and Kim, 2001] deal with segmental feature selection.

8.2.4 Discussion

In [Rhee, Cho and Kim, 2001] it is stated that the segmental feature selection is the most preferable, since it combines the benefits of both the local and global feature selection methods. However, the reached EER (Equal Error Rate) is 3.4 %, while in [Richiardi and Drygajlo, 2003] this number can be 1.7 % using local feature selection. On the other hand, in [Dolfing, Aarts and Oosterhout, 1998] a segmental feature selection method is used, while achieving an EER between 1% and 1.9%. The main difference is that in the latter research thesis the Hidden Markov Model is applied after the selection method to classify the signature instances. In the former case, no classification model is applied.
In [Lee, Berger and Aviczer, 1996] a global feature selection method is used and yields an EER of approximately 5 %. This is higher than the other two selection methods. In [Kholmatov, 2003], [Nalwa, 1997] and [Richiardi, Ketabdar and Drygajlo, 2005], however, it is agreed that local feature selection is more robust and discriminating than global feature selection.

What can be concluded from these studies, is that local feature selection will result in the lowest EER, but will consume more time to complete than global and segmental feature selection, simply because it is more complex and when increasing the signature length, features chosen and the amount of signatures in the database, the time to complete will increase as well compared to the other two selection methods. Fortunately, feature selection has only to be carried out once, when storing the training signatures in the database. When a user wants to verify a signature the feature selection method only has to run on the user's signature that has been provided by the user at that time. Since the purpose of this project is to yield the lowest EER and improve this result, local feature selection will be used.
The realization of local feature selection can be automated. In this project an application written in C#, using the ASP.NET framework, is chosen. The main reason for using this technology is that all processes can be run online over distance via a web interface, since it is preferred that verifying a consignment note is location independent.

8.3 Extreme Points Warping Variant (EPWV)

As described earlier, this technique scans the coordinates of a signature for local minima and maxima. In this way, the movement of a signature is stored and can be compared against the movement of another signature at a later stage.

8.3.1 Preprocessing

In the previous sections it was mentioned how signature data can be converted to a readable format. Data preprocessing was also considered. In this section the realization of data preprocessing will be explained.
Using EPWV, data preprocessing not only consists of filtering out irrelevant data. In addition a second process is needed that converts the filtered data to its string representation. These processes are shown in the figure below.

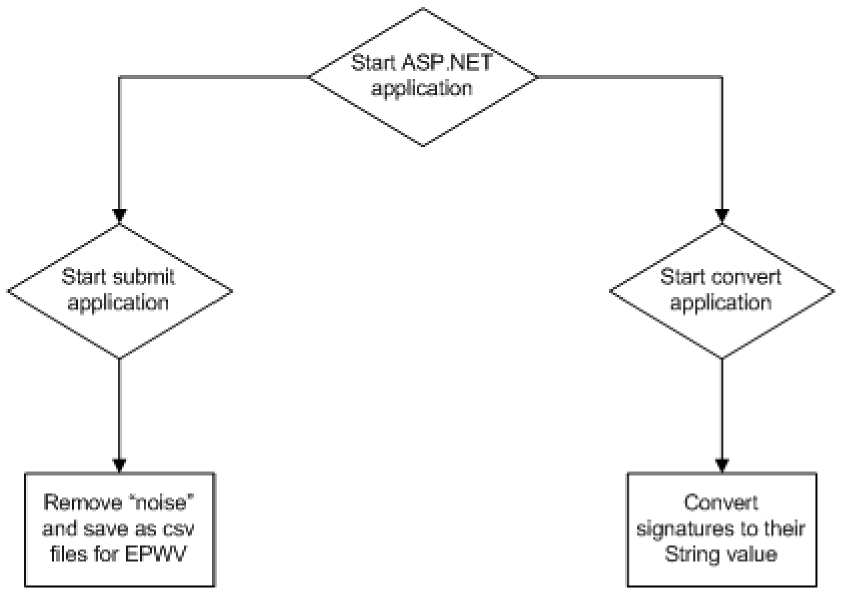

Start ASP.NET application
Start submit application
Start convert application
Remove "noise" and save as csv files for EPWV
Convert signatures to their String value

Figure 8-5: Preparing the data for EPWV

The symbols used in Figure 8-5 have the same meaning as those used in Figure 8-1. First the ASP.NET application is started. This application is capable of executing all subsequent processes automatically. However, in this setup the choice has been made to split the main ASP.NET application initially into three different sub applications. A screenshot of the main ASP.NET application is showed in the figure below.

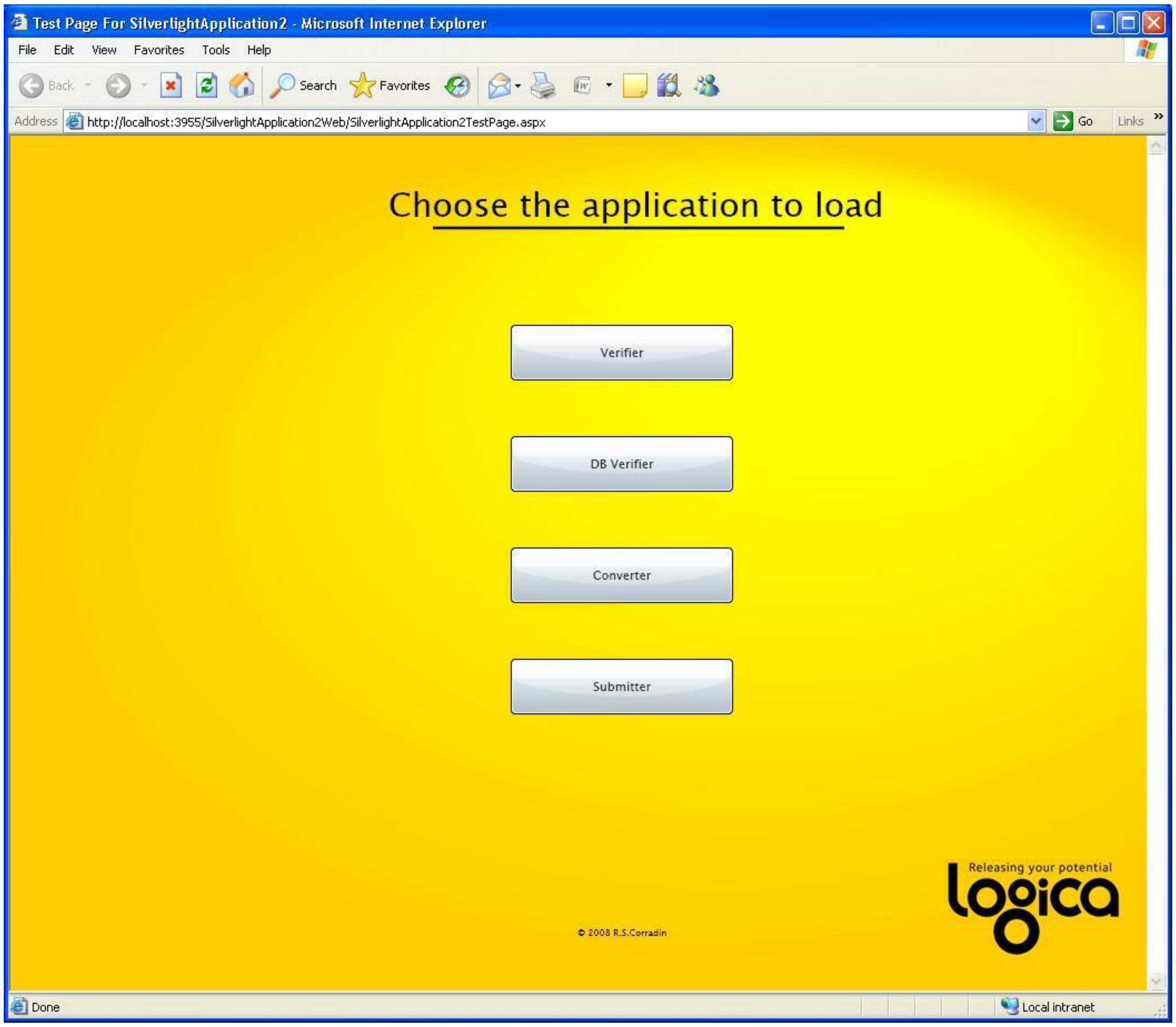
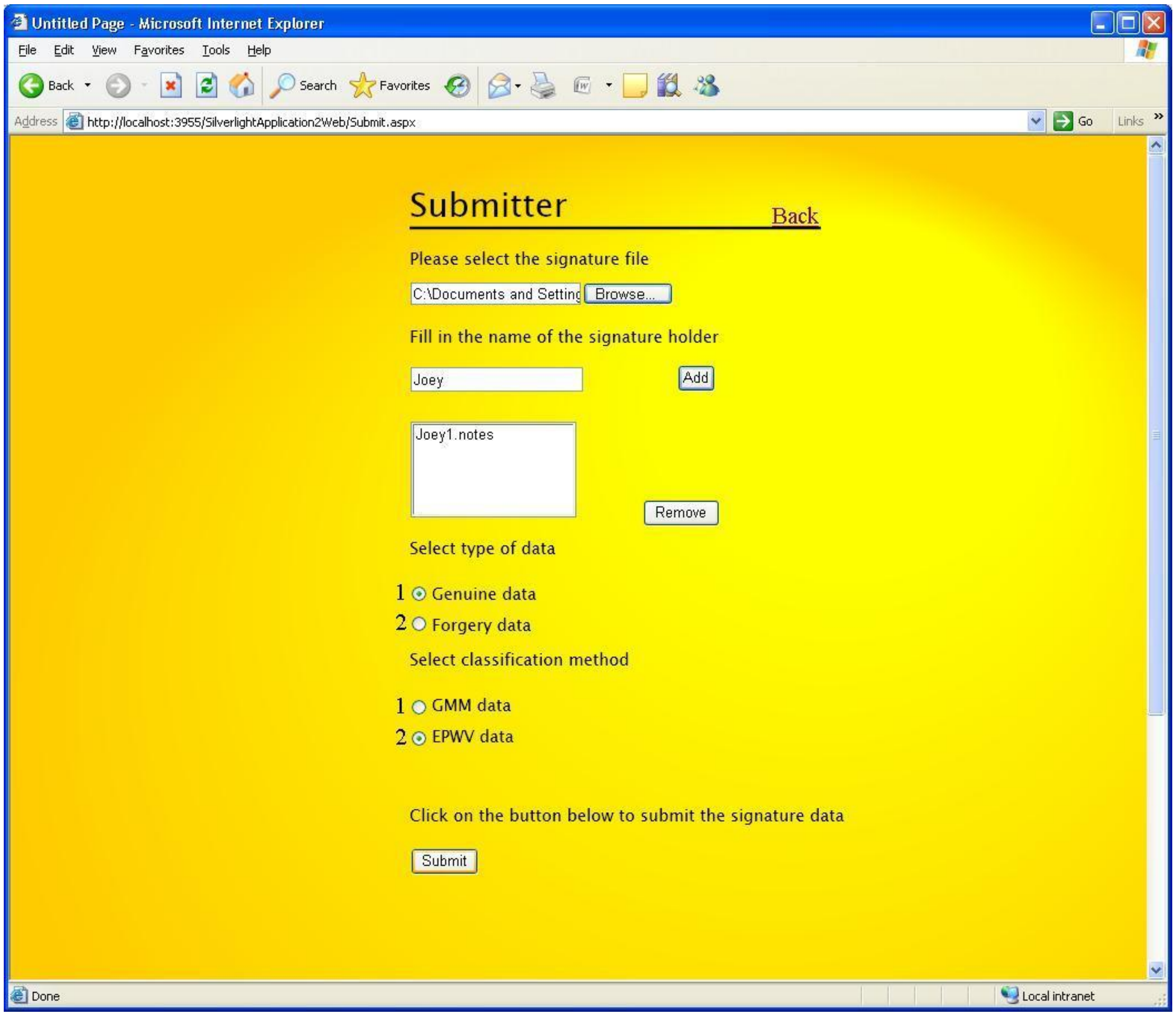

Test Page For SilverlightApplication2 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:3955/SilverlightApplication2Web/SilverlightApplication2TestPage.aspx
Go Links
Choose the application to load
Verifier
DB Verifier
Converter
Submitter
© 2008 R.S.Corradin
Releasing your potential
logica
Done
Local intranet

Figure 8-6: Screenshot of the main application

As depicted above, the application is initiated and the user can then select which sub application to load. As can be seen, there are four applications available instead of three. The fourth application is the Verifier and has been added as an extra feature to verify the signature of a specific user. This application has not been used to determine the FARs FRRs and EERs, but is used for the demonstration of the prototype. The sub applications each represent a particular process in the verification of signatures as will be further explained.

With the submit application on the left side of Figure 8-5 defined earlier, a user can add signatures (generated with MyScript Notes) to a list, which on its turn can be uploaded. This behavior is best illustrated with a screenshot from the submit application.


Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:3955/SilverlightApplication2Web/Submit.aspx
Go Links
Done
Local intranet

Submitter
Back

Please select the signature file

C:\Documents and Setting | Browse...

Fill in the name of the signature holder

Joey | Add

Joey1.notes

Remove

Select type of data

1 [✓ Genuine data]
2 Forgery data

Select classification method

1 GMM data
2 [✓ EPWV data]

Click on the button below to submit the signature data

Submit

Figure 8-7: Screenshot of submit application

As can be seen, a user can browse for the signature file to be submitted and specify the name of the signature holder. Next, after adding the signature to a list (this is done to support the submission of multiple uploads), the user can indicate of what type the signatures are (genuine or forgery) and the verification technique for which the signatures are used (GMM or EPWV). However, the user has to separate genuine and forgery signatures. I.e.: First all genuine signatures have to be submitted; second the forgeries can be submitted.
The files that have to be submitted contain, among the coordinates, other data that is of no use for the verification technique as discussed earlier in section 5.7. The signature files are read one by one and according to the verification technique chosen (EPWV or GMM), the application removes “noise” from the data and stores only relevant data for the chosen technique, as a CSV (comma separated value) file. The conversion to a CSV file is done, because in this way the files can easily be opened in most statistics programs, such as Microsoft Excel.

The genuine and forgery signature CSV files are stored in different folders in order to keep the signature types separate from a user point of view. This eases the implementation from a programmer's point of view as well.

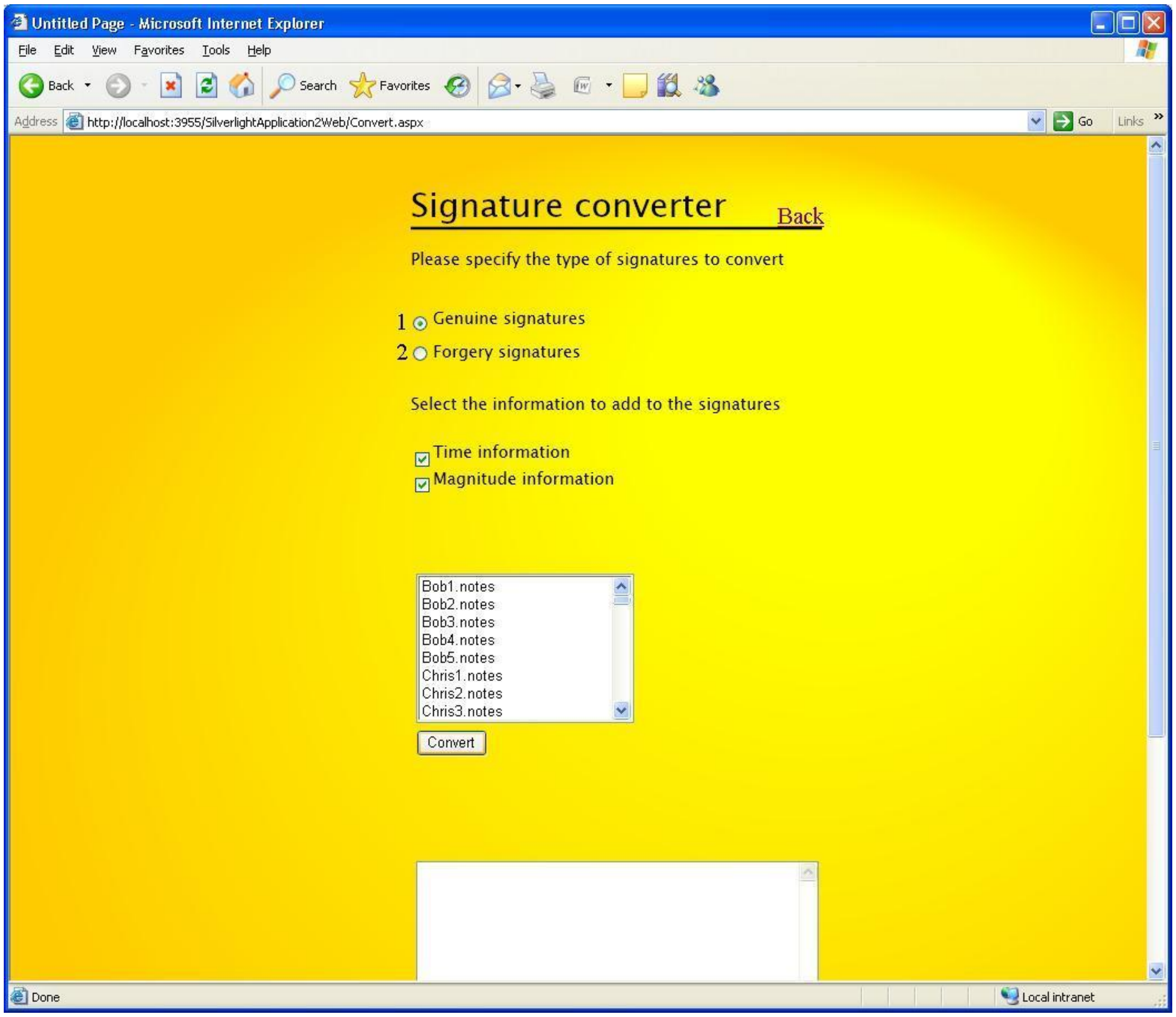
A short trail of the CSV file is shown below in Table III.

Chris
x,y
4280877195 3 3 0
0,0
-7,6
-15,8
-27,6

Table III: Trail of csv file for the EPWV technique

The CSV file begins with the name of the person who signed the signature. In this way, identification is easier and if, for example, a user signs with the name "Chris", only the CSV files that have "Chris" in the first row are checked. This speeds up performance.
The second row displays the type name of the coordinates. In this case, the x and y coordinates. The third row defines a pen lift, as specified earlier in section 5.7.
Finally, the rows that follow are the x and y coordinates, separated by a comma.

While the submit application can be used for both EPWV and GMM, the other applications are EPWV only. The GMM technique requires another approach as is explained later.
The converter application allows a user to automatically convert signatures to their string representation. Analogous to the submit application, this application will be accompanied with a screenshot as shown below.


Untitled Page - Microsoft Internet Explorer

http://localhost:3955/SilverlightApplication2Web/Convert.aspx

Signature converter Back

Please specify the type of signatures to convert

1 [✓] Genuine signatures
2 Forgery signatures

Select the information to add to the signatures

[✓] Time information
[✓] Magnitude information

Bob1.notes
Bob2.notes
Bob3.notes
Bob4.notes
Bob5.notes
Chris1.notes
Chris2.notes
Chris3.notes

Convert

Figure 8-8: Screenshot of convert application

As can be seen, the user can choose the signatures type to load. For example, if “Genuines” is chosen, all genuine CSV files are loaded (since the exact location of these files is known). These CSV files are scanned one by one automatically and each coordinate pair is examined. During this examination, minimum and maximum points are logged as well as the pen lifts.
The user can, furthermore, choose to add features. In this experiment, only the additions of magnitudes and the length of time gaps between extreme points are supported. These are improvements 1 and 3 as described in section 7.2.3.
Finally, the signatures that are present in the list in Figure 8-8 are converted to their string representation and stored in a new folder that contains one CSV file per converted signature. An additional CSV file is created, which contains all converted signatures, one string representation of a signature on each row.

An example trail of this file is shown in the table below.

Bob,BDCDATCPPBDCATBADTTBTCPPACBTDTATBTCTTPPBCDCATDTBTCPPAACBADBCTATBDTTATPPACBTDTTCTDCATTTTDTPPBDACTBTATDTTTBTATCTT
Chris,BDACBTTDTTATTCTBDACBDCATBDTCATBDTCATBATBDTTTATCTBTDTCTATBTDTAATCT
Co,ACBDTATTCTBATDTTBTCTATDPPBCDCTATTBDTCDTATTCPPADCBTDTTATTBCTPPACBDTATBCTPPBCDTCDAATTCPPBCDCATTBDTCDATCPPACBDTCATTDTBTTTCTTTATTBTATDTTTBC

Table IV: converted CSV file example

As can be seen, the conversion is done in such a way that every signature starts with the name of the signature holder followed by its string representation, separated with a comma. In this way, the option to identify a person based on his or her signature is included (signature identification).

Note that the approach in this experiment first converts all signatures to their string representation, before proceeding to the classification of a signature.
This is different from the method suggested in [Gupta and Joyce, 2007], in which a signature is converted to its string representation each time the classification method is run. This alternative method is chosen, since it will result in a shorter computation time when the data set is larger. This is, because the signatures have to be converted only once, rather than every time a test signature is compared against the signatures in the database.

8.3.2 Signature conversion

Continuing the discussion on the conversion of a signature to its string representation, this subject may need some additional explanation, since it is not a trivial one.
To clarify the problem of finding the string representation of a signature the following figure is displayed.

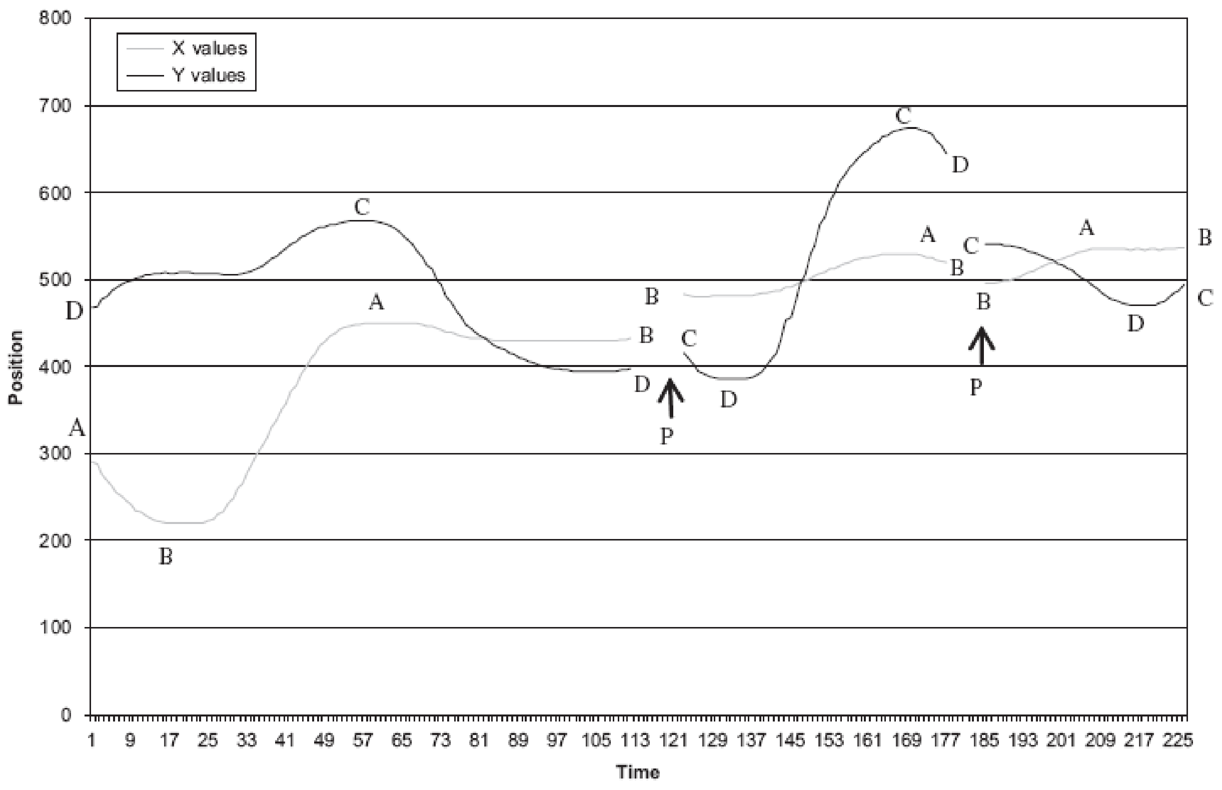


Figure 8-9: x and y values of a partial signature

This figure shows possible x and y values of a small part of a signature as a function of time. This picture is the same as originally defined in [Gupta and Joyce, 2007]. Here, the extreme values are tagged with a specific character. The characters “A” and “C” are used for maximum values of the x and y trajectories respectively. The character “B” and “D”, on the other hand, are used for minimum values of the x and y trajectories respectively. The character “P” represents a pen lift.
The theory is straightforward, but when implementing an algorithm that converts trajectories successfully to their string representation additional problems occur.
We divide these problems as follows:

- Determining an extreme point
- Determining a pen lift

Determining an extreme point

Determining an extreme point may seem straightforward. The following conditional statements can be defined:

$$\max(i) = value(i-1) \ < \ value(i) \ \&\& \ value(i+1) \ < value(i) \tag{9}$$

$$\min(i) = \ value(i-1) > \ value(i) \ \&\& \ value(i+1) \ > value(i) \tag{10}$$

Both statements look at the previous and the next coordinates with respect to time. If those are greater or smaller than the current coordinate a maximum and minimum is found respectively. Unfortunately, many additional cases can be distinguished for which the value of (9) and (10) is undefined. For example, consider what happens if an extreme point is surrounded by coordinates of the same value. According to the statements given earlier, a decision cannot be made. Therefore the algorithm has to look further than only the previous and next coordinate. However, to have an efficient algorithm, this only has to be done when multiple identical values are encountered. This is just one example of when statements (9) and (10) fail in determining an extreme point. Next, additional cases will be listed accompanied by an illustrative example.

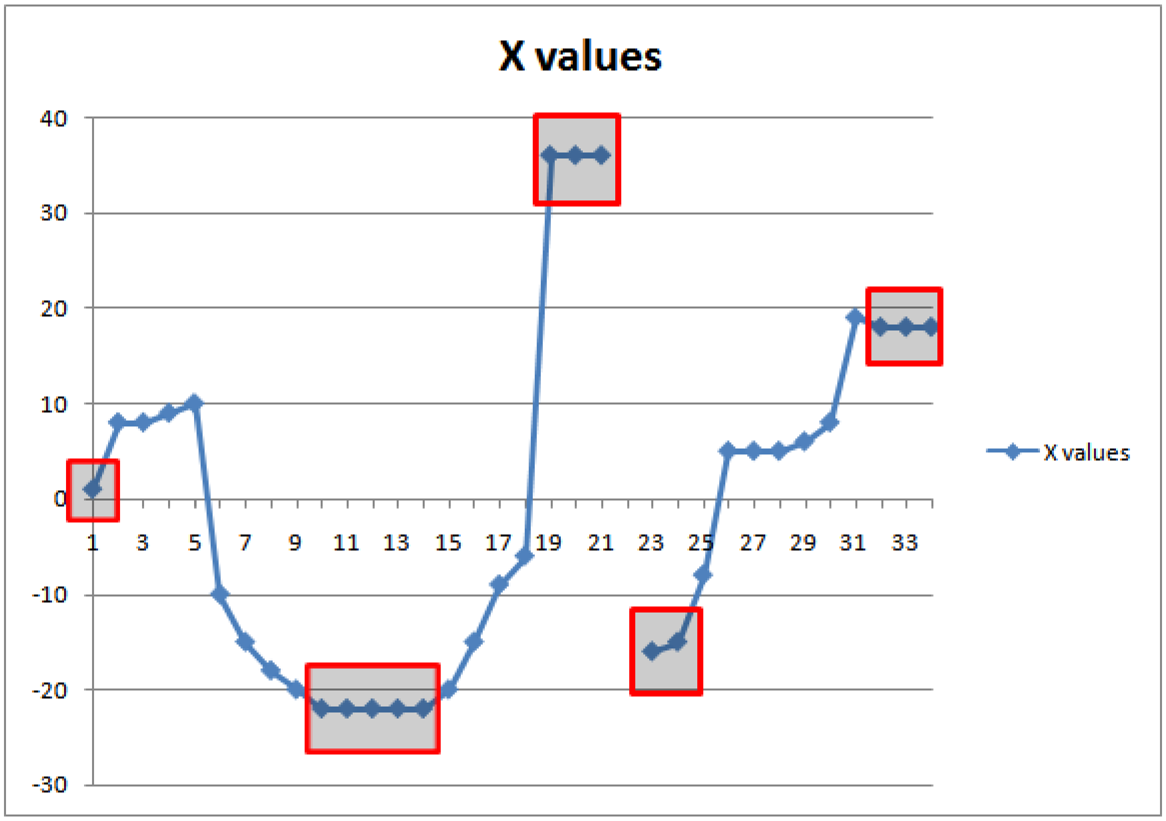


Figure 8-10: x values of a signature example

In Figure 8-10, five red rectangles are drawn, which represent additional checks that have to be made in order to successfully determine an extreme point.
Starting from left to right we can define the following cases:

Determining an extreme point:

1. At the beginning.
2. In between two pen lifts.
3. Before a pen lift.
4. After a pen lift.
5. At the end.

Note that for each red rectangle two scenarios can be defined additionally:

I. Previous and next values differ.
II. Previous or next value is identical.

1. Determining an extreme point at the beginning

The first problem is found at the start of a signature. Assume the signature in Figure 8-10 is not a fragment but represents the whole signature. Then the first red rectangle represents the first coordinate that is registered.
In the case of scenario I, the next value is checked and the extreme points are defined by:

$$\max(i) = value(i+1) < value(i), \qquad with\ i = 0 \tag{11}$$

$$\min(i) = value(i+1) > value(i), \qquad with\ i = 0 \tag{12}$$

For scenario II, “i + 1” has to be incremented until the first non equal coordinate is found in the coordinate list. When this coordinate is found, an extreme point can be determined according to the statements (11) and (12). If all coordinates in the whole signature are identical, then the algorithm cannot determine a maximum or minimum. This behavior is correct, since mathematically this is also not decidable.

2. Determining an extreme point in between two pen lifts

Since every signature is characterized by at least one pen down and one pen up, determining an extreme point, in between, can be considered as a second case. Analogous to the previous case we take into consideration the two scenarios.
The first scenario is straightforward and the statements (9) and (10) result in a correct result. However, for the second scenario (which is illustrated in the second red rectangle in Figure 8-10) the index has to be incremented until a non equal value is obtained and the value 15 is found. Note that in this example, i is 10. This means that the minimum will be stored at position 10. When this is done, the algorithm proceeds to position 15, skipping the intermediate positions. In this algorithm, the first position is taken, since this is the start of a minimum or maximum. This gives a fair comparison, because all starting points of minima and maxima are compared to each other.

The process of determining an extreme point in between two pen lifts is exactly the same as in the previous case. Therefore the two cases can share this algorithm of determining the first non equal index as seen from the current position of the coordinate.

3. Determining an extreme point before a pen lift

Consider the case in which a signature has more than two (one at the beginning and one at the end) pen lifts as is displayed in the third red rectangle in Figure 8-10. This can be seen as determining an extreme point at the end. In fact, for scenario I, it is exactly the same. This also holds for scenario II.
However, in this research, a slightly different algorithm is developed. Consider what happens (scenario II worst case) if all x coordinates before a pen lift are of value, for example, 3. After the pen lift, the first coordinate to follow is of value 4. Then, the algorithm suggested here, classifies the part of the signature before the pen lift as a minimum (the character “B”), while the one in [Gupta and Joyce, 2007] results in an undefined character. This does not mean that the algorithm in [Gupta and Joyce, 2007] is wrong. It is merely interpreted in a different way.

4. Determining an extreme point after a pen lift

The same discussion arises here as in the previous case and corresponds to the fourth red rectangle in Figure 8-10. The algorithm used is equal to determining an extreme point at the beginning for both scenarios, with the difference that the value of i corresponds to the beginning of the pen lift instead of the beginning of the signature.

5. Determining an extreme point at the end

The last case (shown in the fifth red rectangle in Figure 8-10), performs as follows when applying scenario I:

$$\max(i) = value(i-1) < value(i) \quad (13)$$

$$\min(i) = value(i-1) > value(i) \quad (14)$$

For scenario II, a problem may arise. Looking again at the figure, it can be concluded that the three last values are identical. It can also be concluded that the three last values have to be converted to “B”, since it is a minimum. Suppose $i = 32$, referring to the third last coordinate (which is different than the three last ones).
What happens in the algorithm is the following:

- Value of current x coordinate = 18.
- Value of next x coordinate = 18.
- Find next value which is not equal to 18.

The algorithm has a problem in this case and is unable to find the next non equal value, since it does not exist. For this reason, the algorithm is changed in such a way that when it reaches the end of the coordinate list without finding a non equal value, it returns the previous value. In this case the previous value is 19. From this, an extreme point can be determined, namely “B”. Furthermore, since the algorithm knows the end of the file has been reached, it can stop looking further and puts out the total result string.

Determining a pen lift

Determining a pen lift is fairly easy, since from the CSV file it is characterized by its length and number of spaces (this is always the same). If such a row is encountered the character “P” is immediately added to the result string.

8.3.3 Classification

The experiment contains different phases, which are displayed in Figure 8-11 below for clarity.

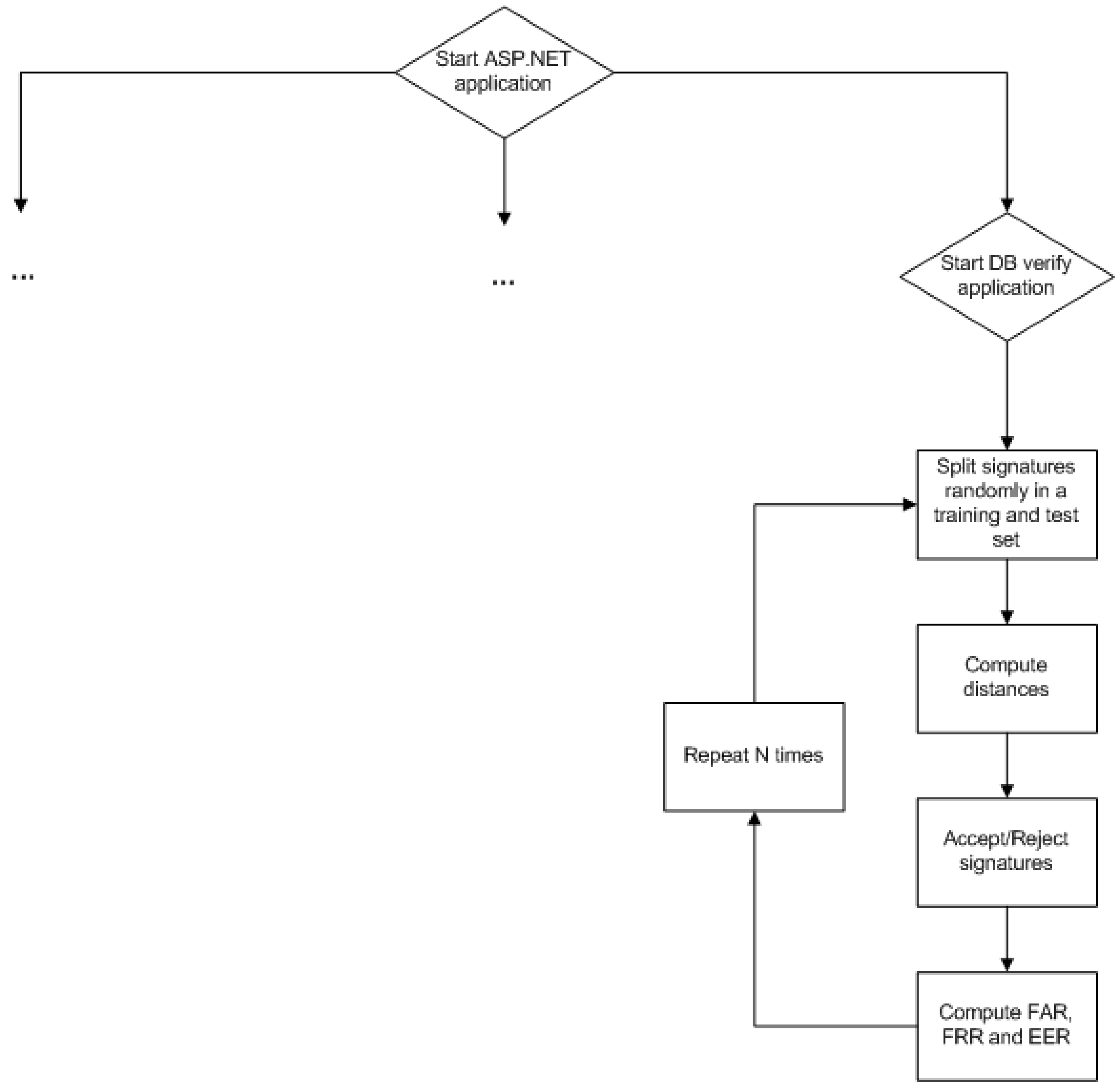

Start ASP.NET application
...
...
Start DB verify application
Split signatures randomly in a training and test set
Compute distances
Repeat N times
Accept/Reject signatures
Compute FAR, FRR and EER

Figure 8-11: EPWV classification experiment steps

The processes displayed in the figure follow those that are displayed in Figure 8-1 and Figure 8-5. I.e.: It is assumed that signatures have been converted to their string representation (as can be derived from the dotted patterns in Figure 8-11).

The “DB verify” application is responsible for verifying all converted signatures and a screenshot of the application is shown below. To train the extreme points warping variant (EPWV) technique, the genuine signature set is split into a training and test set. In this experiment a training set of 4 signatures and a test set of 1 signature per person are used. This process is randomized, i.e.: every time the application is run, 4 signatures are taken randomly from the 5 genuine signatures and the remaining signature is used for the test set. Note that the forgery set is also a test set, since using forgeries as a train set will result high error rates. 2 forgeries per person are created, as described earlier. The application is implemented in such a way that it does not know which test set contains forgeries and which test set contains genuine signatures. By organizing the forgeries and genuine signatures into different file directories, after the classification, the application is told which signatures are genuine or forged. In this way, it is known which signatures are accepted falsely (the forgeries that are accepted) and rejected falsely (the genuine signatures that are rejected).

To compute the distances between signatures the Wagner Fischer algorithm, described in section 7.2.2, is used, which outputs an integer number that represents the distance between two signatures. These outputs will be stored temporarily in a list which represents distances between all the signatures that have been compared against each other. The greater a number in this list, the greater the distance between two strings.

To accept or reject signatures, two types of distances are defined within the application:

1. Distances between genuine training signatures
2. Distances between test signatures and training signatures

The first type is for training purposes and computes the distances between the genuine training signatures per person. When this is done, the mean and standard deviation is calculated and stored per user. This mean and standard deviation is used to determine the global threshold for which the signature is accepted or rejected.
The second type of distance takes the mean of the distance between a test signature (this can be either a genuine or forgery one) and the training signatures. In this case the training size is 2, so each test signature per user (three in total) is compared to two training signatures. Finally the mean of these two distances is taken and compared with the threshold. If the mean distance is smaller, the signature is accepted, otherwise it is rejected. This procedure is executed for every test signature.

To determine if a signature is accepted or rejected, the following condition is used:

$$\mu_{Test} < \mu_{Train} + (Threshold \times \sigma_{Train}) \qquad Accept \tag{15}$$

$$\mu_{Test} \geq \mu_{Train} + (Threshold \times \sigma_{Train}) \qquad Reject$$

μ_{Test} is the mean of the distances between a test signature and the genuine signatures. The second parameter μ_{Train} is the distance mean that is measured from the training signatures to each other. The threshold is a numerical value which according to [Gupta and Joyce, 2007] proves to be most effective (yielding low false acceptances and rejections) between the values 1.5 and 2.0. Finally, σ_{Train} is the standard deviation from the means of the genuine signatures.

Condition (15) can be formulated in words as follows: if the mean of the test distances is smaller than the training distance mean plus the threshold times the standard training deviation, the signature is accepted. Otherwise, it is rejected. The results of this technique vary according to the threshold selected as can be seen in a later section.

The formula from (15) is evaluated and different counters are maintained to calculate the total False Acceptance Rate (FAR) and False Rejection Rate (FRR). Since the application is told in this phase if a signature is genuine or forged, it can determine the FAR and FRR. If a genuine signature is accepted nothing happens. However, if it is rejected a counter that indicates the falsely rejected signatures is incremented. This is done as well for the falsely accepted signatures.

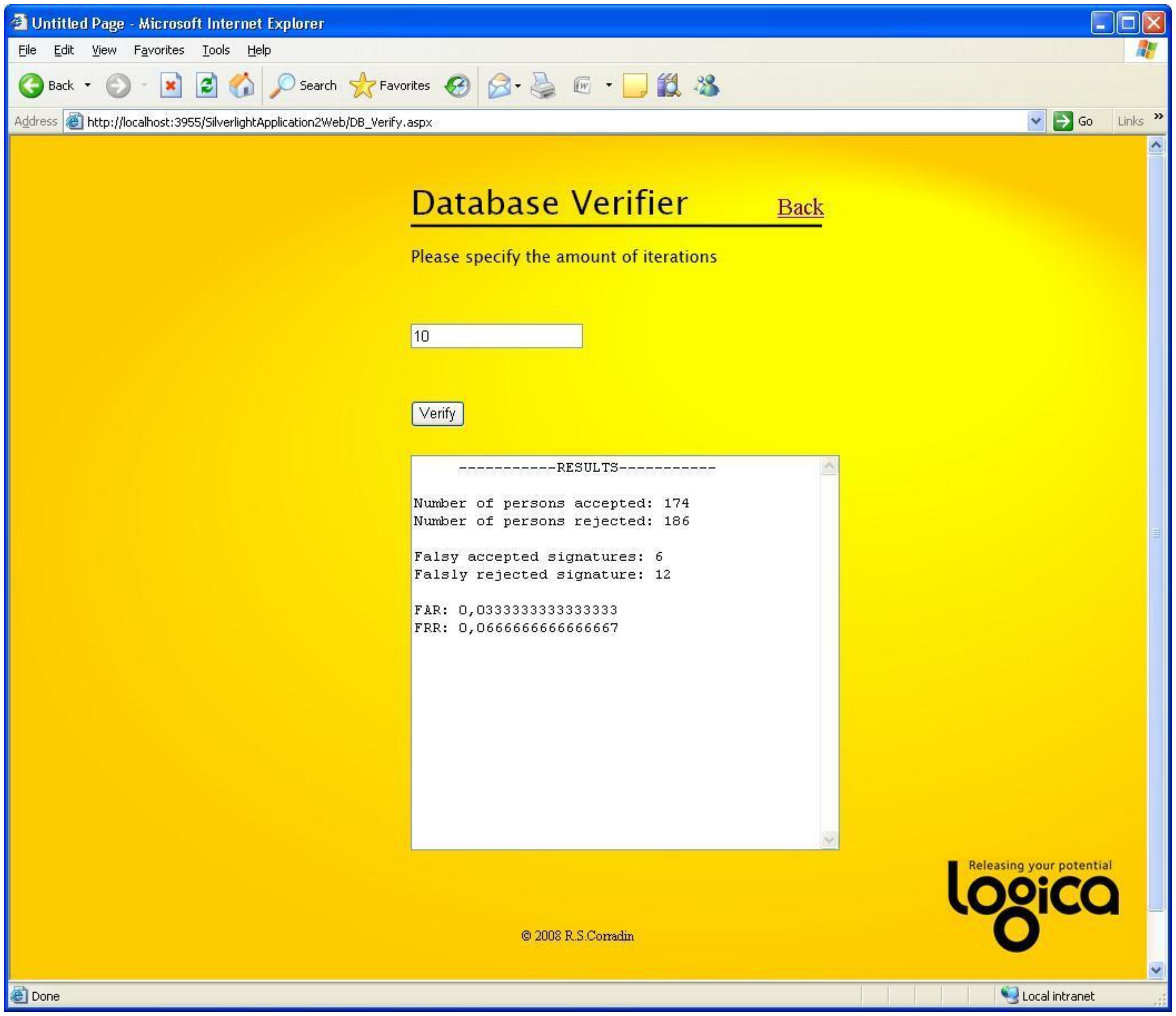
Finally, the FAR and FRR can be calculated by means of the following formulas:

$$FAR = \frac{COUNTER_{FAR}}{TOTAL} \times 100 \tag{16}$$

$$FRR = \frac{COUNTER_{FRR}}{TOTAL} \times 100 \tag{17}$$

In which the counter contains the amount of falsely accepted and rejected signatures as specified earlier. The variable $TOTAL$ contains the total amount of signatures present. Out of these values the equal error rate (EER) can be obtained if desired and is the value of the FAR and FRR when they are both equal to each other.

As an additional option, a user can start the application and is allowed to fill in an iteration value. This iteration value specifies the amount of times the experiment is repeated as shown in Figure 8-11. To illustrate what the addition of this feature means for a user, the screenshot of the “DB verify” application is shown below.


Untitled Page - Microsoft Internet Explorer
http://localhost:3955/SilverlightApplication2Web/DB_Verify.aspx
Database Verifier
Back
Please specify the amount of iterations
10
Verify
Releasing your potential
© 2008 R.S.Corradin
Done
Local intranet

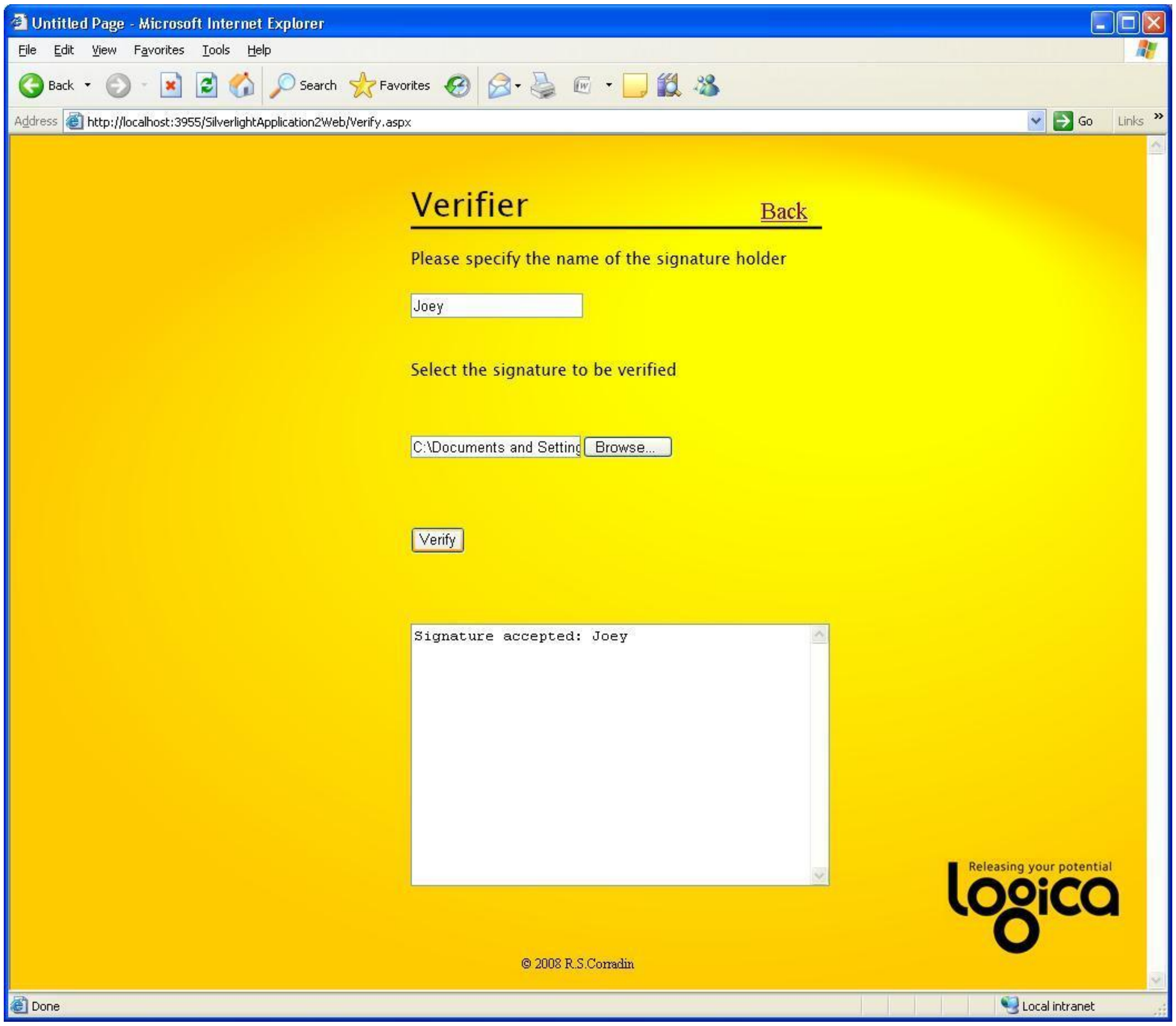
```
-----------RESULTS-----------

Number of persons accepted: 174
Number of persons rejected: 186

Falsy accepted signatures: 6
Falsly rejected signature: 12

FAR: 0,0333333333333333
FRR: 0,0666666666666667
```

Figure 8-12: Screenshot of DB verify application

As can be seen from the screenshot, the iteration value is shown at the top of the page. This iteration value specifies the amount of times the verification process is executed. Since the process of selecting a training and test set is randomized, different outcomes are possible each time the verification process is run.
In the above figure, the verification process is run 10 times. Since one iteration checks 18 signatures, 10 iterations will check a total of 180 signatures. 10 iterations, however will give a result, which is more reliable than a result generated by 1 run, since the mean result of all iterations is taken. In the above figure a total of 174 + 186 = 360 signatures are checked.

This is because the process includes 180 times checking the genuine and 180 times checking the forgery signatures.

Before, it was mentioned that the main ASP.NET application consists of four sub applications. The fourth application is the verifier and is added as extra functionality for a user that would like to check if one input signature is recognized. This is shown in the figure below.


Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:3955/SilverlightApplication2Web/Verify.aspx
Go Links
Verifier
Back
Please specify the name of the signature holder
Joey
Select the signature to be verified
C:\Documents and Setting
Browse...
Verify
Releasing your potential
logica
© 2008 R.S.Corradin
Done
Local intranet

```
Signature accepted: Joey
```

Figure 8-13: Screenshot of verify application

As displayed in the figure, the name of the signature holder can be specified and the corresponding signature file (this is the file converted by MyScript Notes) can be added. After the user clicks on the “Verify” button the application will use the EPWV classification algorithm to determine if the signature is accepted or rejected. The outcome will be printed in the text field below the “Verify” button.

8.3.4 Results

After having run the application multiple times (altering parameters of each run), the FARs and FRRs are gathered. It is again emphasized that the EPWV experiment follows the steps described in [Gupta and Joyce, 2007] closely. The main difference between the experiment set-up of the EPWV technique and the GMM technique is that a randomized training set is chosen every time the application is run. This could mean that if there are 10 iterations, multiple equal outcomes are yielded, since randomization can pick the same training set twice (or more). In fact in this experiment 10 iterations are used, together with a training size of 4 and a test size of 1. The forgery size is always 2, since each person tries to mimic a signature of another person twice.
The results of the EPWV technique are summarized in the table below.

	Signatures accepted	Signatures rejected	Signatures accepted falsely	Signatures rejected falsely	Threshold	FAR	FRR	EER
All features	179	181	16	17	1.77	0.09	0.09	**0.09**
Only magnitudes	182	178	20	18	1.45	0.11	0.1	**0.1**
Only Time info	180	180	16	16	1.85	0.09	0.09	**0.089**
None of the above	179	181	18	19	1.46	0.1	0.11	**0.1**

Table V: Summarized EPWV results

As can be seen from the table four different tests are made.
The first row includes the addition of both magnitudes and time info to a signature. These are improvements 1 and 3 described in 7.2.3. The second and third rows are self explanatory. The fourth row contains only the basic EPWV technique without additions.
The columns in the table may require some additional explanation. The first two columns specify the amount of signatures accepted or rejected. Since 10 iterations are made in this experiment, a total of 180 signatures are scanned. This is the reason why the numbers in the first four columns may exceed the number of signatures that have been gathered.
The third and fourth columns specify how much signatures are accepted or rejected falsely from the first two columns. For example, looking at the first row 16 of the 179 accepted signatures have been accepted falsely and 17 from the 181 rejected signatures have been rejected falsely. The threshold in the fifth column is the one specified earlier in equation (15). The sixth and seventh column contain the FAR and FRR values. Note that this table is a summarization of the tables displayed in

Appendix D. The threshold is fine tuned to obtain the EER, which is displayed in the last column.

To determine the threshold for which the FAR and FRR are equal the following graph can be plotted.

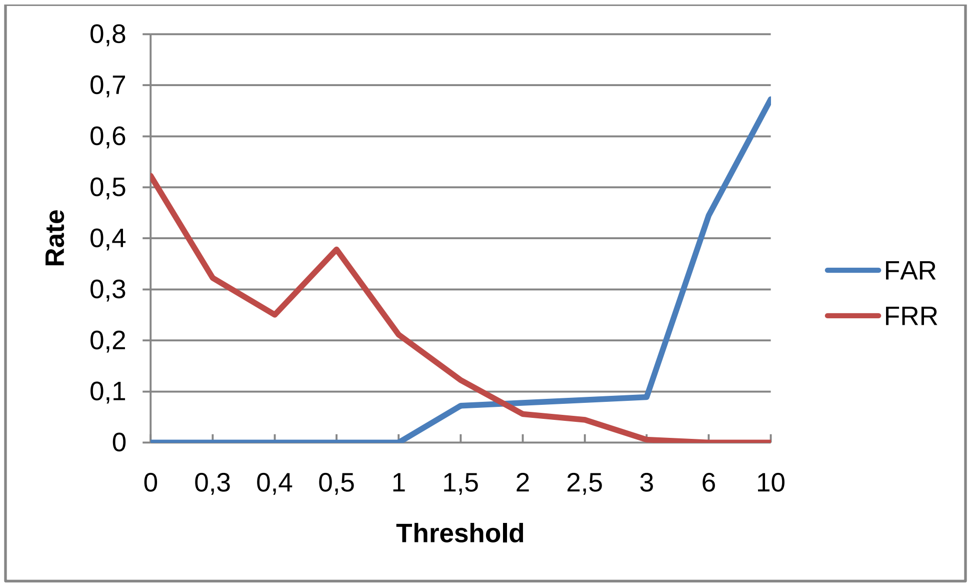


Figure 8-14: FAR and FRR plot

In the above graph, the FARs and FRRs are plotted for the case in which both magnitudes and time information have been added. On the vertical axis the rate (probability between zero and one) is displayed and on the horizontal axis the threshold is shown. The point of interest in this graph is the threshold for which both rates are equal. This is between the values 1.5 and 2 leaning more towards the value 2. In fact, after experimenting with different values in this range, the value that meets these requirements is approximately 1.77. All EERs for the EPWV technique are obtained in this way.

This section is concluded with an evaluation on the results that have been obtained for the EPWV technique.
The results do not differ much from the ones obtained in [Gupta and Joyce, 2007]. In fact, better initial results are obtained. In this experiment only skilled forgeries are considered. In [Gupta and Joyce, 2007] an EER of 14% is obtained in the basic case (without any additions), while in this experiment this value is 10%. This difference could be justified by saying that different datasets with varying size are used. Since in this research a relatively small amount of signatures is gathered, the EER can differ from experiments done with a larger signature set. To continue this discussion, adding magnitudes and time information yields an EER of 9% in this experiment, while in [Gupta and Joyce, 2007] this value is 4.8%. However, in [Gupta and Joyce, 2007] global features (the second improvement in 7.2.3) are added before adding time information. It is expected that the elimination of this addition (in [Gupta and Joyce, 2007]) will result in a higher EER closer to the one in this research.

In general, what can be concluded based on the results is that adding improvements lowers the EER. Adding Time information has a higher influence on lowering the EER than adding magnitudes. This can be justified by saying that less magnitude symbols are added to the string representation of a signature compared to the time information symbols.
All in all, EPWV shows promising results when applied in Anoto environment as has been proven in this experiment.

8.4 Gaussian Mixture Models (GMMs)

In contrast to EPWV, Gaussian mixture models do not need to convert the input coordinates of a signature to a particular format. This saves a lot of time, since the coordinates can be used immediately for classification. Furthermore, the algorithm to classify the signatures does not need to be implemented manually, since general GMM methods already exist and are freely available. The signature data, however, still contains “noise” and needs to be filtered out.

8.4.1 Preprocessing

This filtering is done analogously to the EPWV technique. In fact, the ASP .NET application that is primarily used for the EPWV technique is now used to remove noisy data and prepare csv files for the GMM technique. This is illustrated in the figure shown next.

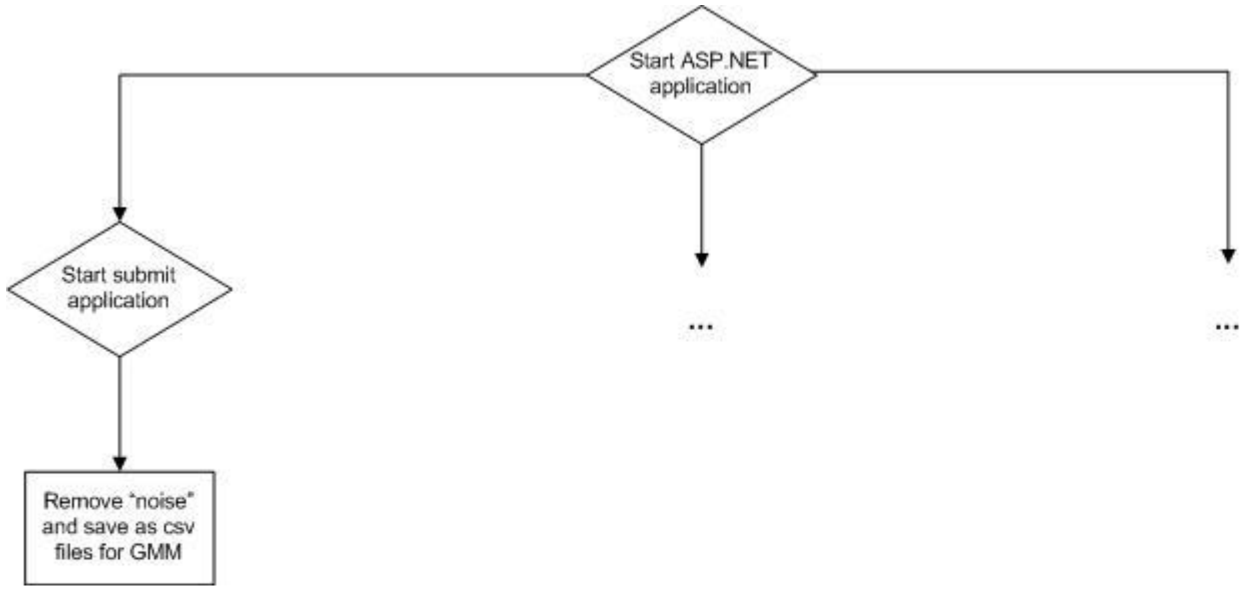

Start ASP.NET application
Start submit application
Remove "noise" and save as csv files for GMM
...
...

Figure 8-15: Submit GMM files

As can be seen from the figure, the ASP.NET application (in particular the submit application) is started and irrelevant data can be discarded. The remainder of the application has been left out of the figure for clarity. As was indicated in section 8.3.1, a user can choose whether to submit the data for the EPWV or GMM technique. In this case, the GMM technique is chosen and the application converts the data to a csv file that contains only information relevant to the GMM technique.

As an example, a short trail of a csv file is displayed below.

Chris
x,y
0,0
-7,6
-15,8
-27,6

Table VI: Trail of csv file for GMM technique

As can be seen, the difference between this table and the earlier defined Table III, is that the second row contains the first coordinate pair, instead of a pen lift. I.e.: Pen lift information has been left out. Of course, in a future approach customization can be done and technique specific features could be added either for the EPWV or the GMM approach. To support this addition of functionality, the application has been implemented to separate the submitting of signatures for different techniques.

8.4.2 Classification

After all signatures have been submitted and the corresponding csv files have been generated, the remainder of this experiment is done in Matlab. The choice for Matlab is made, because it provides excellent support for Gaussian mixture model classification in the form of a toolbox. This toolbox is called NetLab ([Nabney and Bishop, 2001]) and is freely available for Matlab. Furthermore, Matlab provides all the functionality needed to read, manipulate and store csv files.

To identify the different steps in the experiment, Figure 8-16 is defined below.

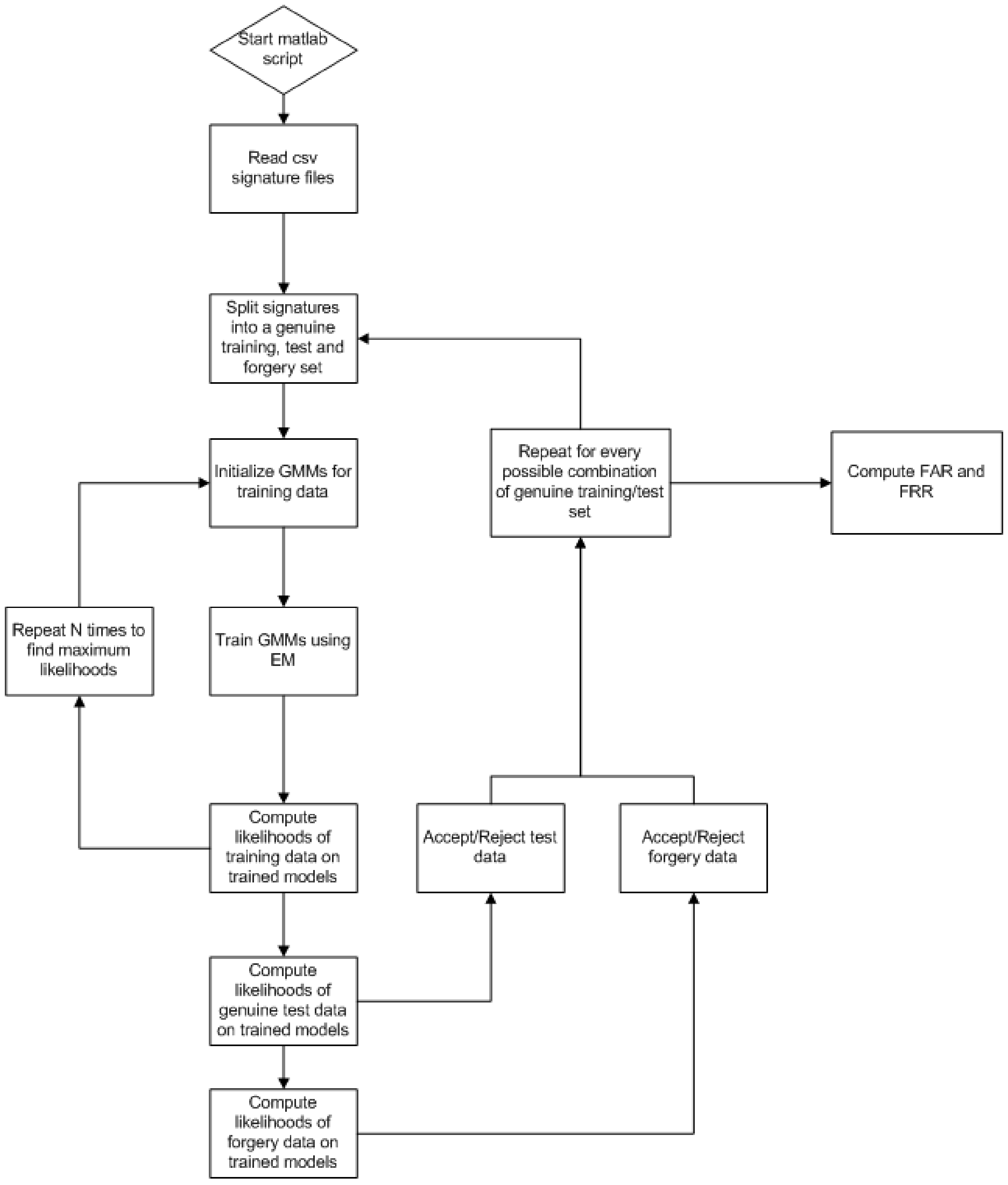

Start matlab script
Read csv signature files
Split signatures into a genuine training, test and forgery set
Initialize GMMs for training data
Repeat for every possible combination of genuine training/test set
Compute FAR and FRR
Repeat N times to find maximum likelihoods
Train GMMs using EM
Compute likelihoods of training data on trained models
Accept/Reject test data
Accept/Reject forgery data
Compute likelihoods of genuine test data on trained models
Compute likelihoods of forgery data on trained models

Figure 8-16: GMM experiment steps

The experiment starts with running the Matlab script, which contains different processes. This is displayed as the diamond symbol in the upper part of the figure. The processes that follow are all automated by the script and will be explained here in more detail.

When initiated, the script reads the csv files one by one. It is made sure the output path of the .NET application is identical to the input path of the Matlab script, otherwise the files cannot be located. Furthermore, since the genuine csv files are split from the forgery csv files, the script knows exactly which files belong to which type of signature. This separation is crucial when determining the FAR and FRR in the final process. The script maintains this separation of signature types during its execution.

Once all signatures have been read, different sets are created. The genuine signatures are split into a training and test set. This is done sequentially, so every time the script is run, a different combination (one that has not been chosen in previous runs) of training and test set is obtained. The forgery signatures are gathered as well and form the second test set (the forgery set) in the verification process. Note that the forgery set is always the same, but since the training set differs every time the script is run, a different outcome is yielded as well for the forgery set.
The main reason for choosing different training and test sets is reliability and completeness. Since a different outcome is yielded for different training and test sets the mean outcome is more realistic and reliable. An outcome based on only one run will not tell much about the general accuracy of the classification technique. Furthermore, applying every combination of training and test set on the classification process covers all possibilities, hence the result is complete.

Next, the GMMs have to be initialized. With initialization, it is meant that a new Gaussian mixture model is created. This model can be created using certain parameters. These parameters have to be known in order to successfully build a GMM. In Matlab, GMM initializtion is split into two functions: creation and initialization. The first function creates a GMM (the structure of the GMM) and the second function initializes this GMM with the signature data that is available. These are two separated functions. The function that is able to create a GMM needs the following parameters to be initialized:

- Dimension
- Number of centers
- Type of covariance matrix

The dimension is simply the dimensionality of the space from which the data points are taken. In this experiment, the dimension is 2, since x and y coordinates are of the same dimension. A signature as a whole is two dimensional.
The number of centers is the same as the Gaussian mixture components. Earlier it was said that according to [Richiardi and Drygajlo, 2003] a value of 16 and 32 yield the best result. Different values will be tried in order to find the best setup in this experiment.
Finally, the type of covariance matrix defines the structure of the each component. For example, a spherical covariance matrix will try to cluster the data based on spheres. To illustrate this consider the following figure.

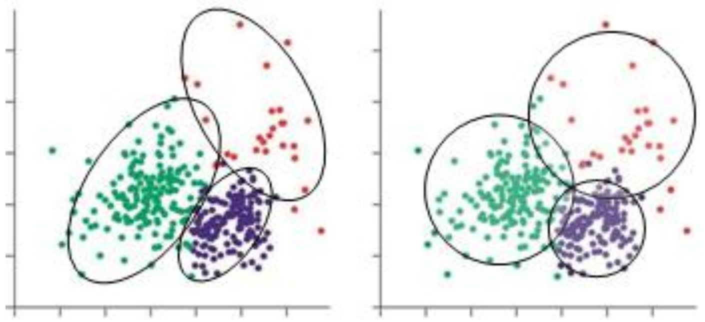


Figure 8-17: GMM covariance example

In the left figure, the covariance type is set to full, which allows classifying the data based on elliptical forms. In the right figure, the type is set to spherical. As with the number of centers, different covariance types are tried and it is examined to what extent they influence the classification process.
The second function in Matlab takes the created GMM (with dimension, number of centers and covariance type) together with the actual signature input data and starts with the initialization procedure. Furthermore, it can be specified how many iterations the K-means algorithm has to make. This algorithm is able of determining the centers of the GMM and the outcome will vary according to the amount of iterations. Analogously to the number of centers and covariance type, the K-means algorithm will have different values initially for experimental reasons. More information about the K-means algorithm can be found in [Jung, Yoon and Kim, 2000] and [Richiardi and Drygajlo, 2003].

The initialized GMM has to be trained in order to perform well on the test set. This training is done using the EM (Expectation Maximization) algorithm, in which the GMM parameters are re-estimated iteratively in order to fit the input data. During these iterations the parameters will converge to values that relate closely to the model of the input data. These parameters are the mean, covariance and the weight for each component in the GMM.
As with the K-means algorithm, a value is needed to define the iterations. The parameters needed for the estimation are:

- The GMM after the K-means initialization
- The input data
- The amount of iterations for EM re-estimation

The first two parameters are of the exact same form as in the initialization process, with the only difference that the first parameter is the GMM after the initialization, instead of before.
The third parameter specifies the amount of iterations for the EM re-estimation. In this experiment it is set to 10, since this value is also used in [Richiardi and Drygajlo, 2003] and has proven to yield satisfactory results. Of course, other values are considered and varying results are measured as well.

Once the GMMs are initialized and trained, the likelihood of the training data is computed. Another approach is to immediately use the test and forgery data as input for the trained models and compare likelihoods. This, however, leads to the problem that it is not known what “good” results are. For example, if a test signature is put in and the accuracy is 65%, how do we know if this is good or bad? This answer cannot be solved, unless it is known how well the original training data performs on the models that have been generated from this data. For this reason the training data is used as input for the Gaussian mixture models and the likelihoods are determined. For this, the formula from equation (8) is used. Applying this formula for process discussed here results in the following:

$$TrPM(i,1) = [\log \mathrm{p}(\mathrm{O}^{\mathrm{Train_i}}|\Theta^{\mathrm{Train_i}}) - \log \mathrm{p}(\mathrm{O}^{\mathrm{Train_i}}|\,\Theta^{\mathrm{Train_{Total-i}}})]$$

$With$:

$TrPM, the\ Training\ Probabilities\ Matrix$

$\mathrm{O}^{\mathrm{Train_i}}, the\ training\ input\ data\ of\ a\ signature\ from\ person\ i$

$\Theta^{\mathrm{Train_i}}, the\ training\ data\ model\ for\ person\ i$

$\Theta^{\mathrm{Train_{Total-i}}}, the\ training\ data\ model\ for\ all\ persons\ except\ person\ i$

As can be seen, the likelihood that a signature belongs to a specific person is calculated. In the Matlab program, only likelihoods between zero and one are taken and stored into the matrix, instead of taking the log function, but the theory is the same.
Furthermore, the likelihoods are stored in such a way that they later can be compared against the test and forgery likelihoods.
This formula has to be integrated in the Matlab program to consider all input signatures and signature models. The following algorithm can be defined:

```
for(int i = 0; i < sizeof(training_models) i++)
{
        for(int j = i; j < sizeof(training_set_size); j++)
        {
                TrPM(i,j) = log p(train_signature(i * training_set_size) + j - 1) |
                                training_models(i)) –
                                log p(train_signature(i * training_set_size) + j – 1)
                                | training_models(Total - i));
        }
}
```

In short, for a specific model, a training input is taken (the input from one signature). For this input the likelihoods, that it belongs to the model that was generated using that input, is calculated. This is done for all signatures that are generated by a specific model. In this experiment, this means that four probabilities between zero and one are generated for each model, since the training set consists of four signatures.

To clarify this process, consider the following example: 4 persons sign 5 times, creating a genuine signature set of size 20. The training size is 4 signatures per person, creating a training set of size 16. Since there are 4 persons, 4 models are created and the result of the above algorithm becomes:

$$TrPM = \begin{bmatrix} 0.70 & 0.75 & 0.69 & 0.72 \\ 0.80 & 0.75 & 0.78 & 0.70 \\ 0.60 & 0.66 & 0.65 & 0.60 \\ 0.81 & 0.81 & 0.79 & 0.78 \end{bmatrix}$$

Each row represents the model of a person, so the first row respresents the model for the first person. The columns represent the training size. As mentioned earlier, likelihoods between zero and one are calculated by the Matlab program instead of the log likelihood.
Of each row, the mean is taken and is considered to be a “good” outcome. This results in the following list:

$$\overline{TrPM} = \begin{bmatrix} 0.72 \\ 0{,}76 \\ 0{,}63 \\ 0{,}80 \end{bmatrix}$$

In this example there are 4 mean values. In the actual experiment 18 mean values are calculated, since the dataset consists of 18 different persons and models.
The mean values are needed to eventually determine which signatures are accepted or rejected. Note that the algorithm only checks the signatures of the model they belong to, instead of checking all signatures for every model. This is done to reduce the execution time.

Initializing and training the GMMs is a process that can yield different outcomes each time it is run. This is mainly because of randomization in the initialization of Gaussian mixture modeling. As described in [Koh Chin Wei, 2006] and explained earlier, the GMM technique starts with the initialization of a model, using the K-means algorithm to divide the features into a specific amount of clusters. This is done randomly.
A second reason why different outcomes are yielded is the training of the GMM. It depends on the input data and the initialization of the GMM how fast the EM algorithm converges to a sufficient value. Sometimes this can be after 5 iterations, but there are also cases in which this can be more.
For these reasons, the process of initializing and training a GMM is repeated multiple times and the best likelihood (highest outcome of the EM re-estimation) is chosen and used for further classification. In this experiment N is 4, since this yields good results.

After the likelihoods of the training data are determined, the genuine test data and forgery data is determined next. The likelihood a certain observation is seen given a particular model is calculated. This is done using equation (8) defined earlier in section 7.3.3. Given an input signature, a comparison is made to the corresponding models. This equation is exactly the same as the $TrPM$ equation defined previously, with the only difference that now the test and forgery set is compared to the training models.

For clarity both redefined equations are displayed below. Only new symbols are explained.

$$TPM(i,1) = [\log \mathrm{p}(\mathrm{O}^{\mathrm{Test}_\mathrm{i}}|\Theta^{\mathrm{Train}_\mathrm{i}}) - \log \mathrm{p}(\mathrm{O}^{\mathrm{Test}_\mathrm{i}}|\,\Theta^{\mathrm{Train}_{\mathrm{Total-i}}})]$$
$$With:$$
$$TPM, the\ Test\ Probabilities\ Matrix$$
$$\mathrm{O}^{\mathrm{Test}_\mathrm{i}}, the\ test\ input\ data\ of\ a\ signature\ from\ person\ i$$

And:

$$FPM(i,1) = [\log \mathrm{p}(\mathrm{O}^{\mathrm{Forgery}_\mathrm{i}}|\Theta^{\mathrm{Train}_\mathrm{i}}) - \log \mathrm{p}(\mathrm{O}^{\mathrm{Forgery}_\mathrm{i}}|\,\Theta^{\mathrm{Train}_{\mathrm{Total-i}}})]$$
$$With:$$
$$FPM, the\ Forgery\ Probabilities\ Matrix$$
$$\mathrm{O}^{\mathrm{Forgery}_\mathrm{i}}, the\ forgery\ input\ data\ of\ a\ signature\ from\ person\ i$$

The corresponding algorithms are:

```
for(int i = 0; i < sizeof(training_models) i++)
{
        for(int j = i; j < test_set_size; j++)
        {
                TPM(i,j) = log p(test_signature(i * test_set_size + j - 1) | training_models(i))
                                - log p(test_signature(i * test_set_size + j - 1)
                                | training_models(Total - i));
        }
}
```

```
for(int i = 0; i < sizeof(training_models) i++)
{
        for(int j = i; j < forgery_set_size; j++)
        {
                FPM(i,j) = log p(forgery_signature(i * forgery_set_size + j - 1) |
                                training_models(i))
                                - log p(forgery_signature(i * forgery_set_size + j - 1)
                                | training_models(Total - i));
        }
}
```

Note that in the test and forgery cases, the processes of classification are exactly the same as with the training case. The main difference is the input data (and the eventual results), but the methods to yield these results are analogous to each other. Again, only the input data that belongs to a particular model is checked. Consider the following example. A person who produces a forgery signature claims to be person A. The Matlab program only has to calculate the probability of this input signature given the model for person A. This outcome can be compared with the outcome of the training probabilities as defined earlier to determine if the signature should be accepted or rejected.

Once all likelihoods are obtained, these likelihoods have to be tested on certain rules that specify the acceptance and rejection of a signature. At this stage, the earlier determined training likelihoods are of a crucial importance. To determine if a signature is accepted or rejected, again equation (15) from section 8.3.3 is used.

In the Matlab program, information about which type of signature is fed to the algorithm at a certain time is available. The test set contains only genuine signatures and the forgery set only fake ones.
The challenge is to find the right threshold for which a signature is accepted or rejected. A threshold has to be found in such a way that a low EER can be yielded.
This means that the test and forgery input data with a low likelihood (compared to the training input likelihoods) will be rejected and test and forgery input data with a high likelihood will be accepted. The threshold plays an important role in the definition of “low” and “high”.

Before calculating the FAR and FRR, the experiment repeats itself. As can be seen from Figure 8-16, this means restart and split signatures into trainings and test sets. However, this time a different training and test set is chosen. This process is repeated until every combination of training and test set has been classified. For this experiment the amount of iterations is 5, since 4 signatures are used for training and 1 for testing. This can be done in 5 different ways. Note that this is different from the EPWV technique, in which the iteration process is randomized.

Once all signature data has been processed by the application, the FAR and FRR can be determined by using equations (16) and (17) described in section 8.3.3. The threshold has to be found, however by experimenting and using different values for which the FAR and FRR are both low. Eventually the EER can be calculated, since this is the value for which the FAR and FRR are the same.

8.4.3 Results

The Matlab program is run multiple times with different parameters. As stated before, every time the script is run, 5 iterations are made. The training set consists of 4 genuine signatures per user, the test set 1 genuine signature per user and the forgery set 2 forgery signatures per user. The results of the GMM technique are summarized in the table below.

Components	Covariance	K-means	EM	Threshold	FAR	FRR	EER
8	Diagonal	5	5	1.2	0.1613	0.1684	0.16
16	Diagonal	5	5	2.35	0.1579	0.1505	0.15
24	Diagonal	5	5	3.15	0.1277	0.1277	0.13
32	Diagonal	5	5	3.39	0.1684	0.1613	0.16

Table VII: Summarized GMM results

There are a couple of remarks to be made to the above table. First of all, only “diagonal” is used as covariance type in the second column. More covariance types have been used, such as a full and spherical covariance matrix type. Using a full covariance matrix type did not result in noticeable lower EERs, while the completion time of the program increased significantly. On the other hand, using a spherical covariance matrix type led to an increase of EERs. For these reasons, a diagonal covariance type was chosen.
Increasing the amount of iterations of the K-means and Expectation Maximization (EM) algorithm in the third and fourth column of the table did result in a higher completion time and minor improvement. A value of 5 has been found to produce fairly good results.

Another remark to be made is that, according to the table, 24 components in the first column yield an EER of approximately 0.13 in the last column. In [Richiardi and Drygajlo, 2003], however it is found that 16 and 32 components will eventually result in the lowest EER. It is difficult to indicate the cause for this difference, since a lot of factors can play a role. On the other hand, this confirms the discussion about how a difference in datasets can lead to different results. I.e.: There is no fixed optimum for the number of components that yields the lowest EER for every dataset. This amount has to be determined according to the dataset that is used.

The threshold in the fifth column is adjusted in such a way that an (approximate) equal FAR and FRR are found. This is analogous to the EPWV experiment. From this value the EER can be determined. Again, this table is a summarization of the four tables defined in Appendix E.

For completeness, the corresponding plot is shown below (of the first row in Table VII).

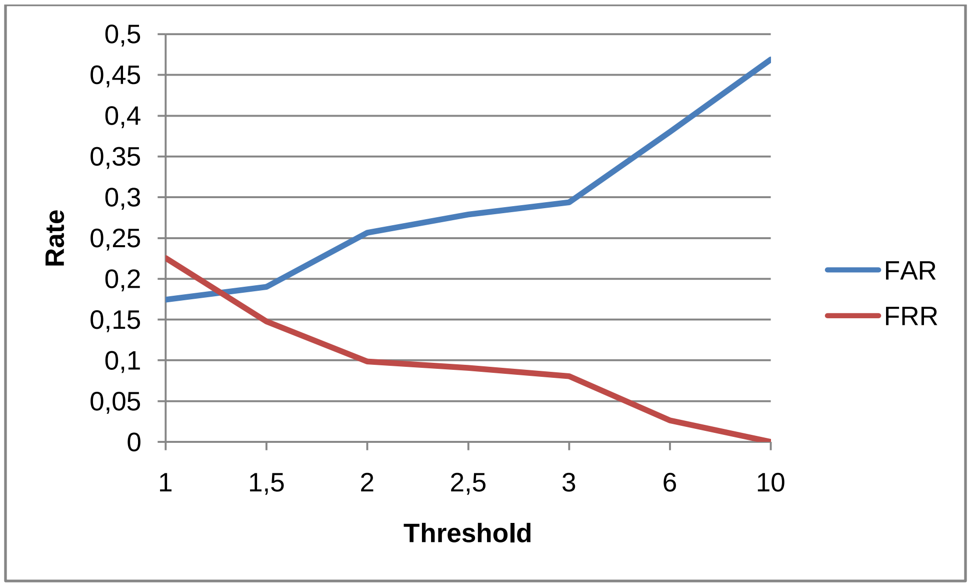


Figure 8-18: FAR and FRR plot

As can be seen from this figure, the EER is found by using a threshold of approximately 1.25. During the experiment, different thresholds have been used in which the value 1.20 was found as the best matching threshold where the FAR and FRR are almost of an equal value.

The results in this section are, compared to [Richiardi and Drygajlo, 2003], significantly higher. As mentioned before, this difference can depend on many factors. The most important ones are the size and type of the signature data. In [Richiardi and Drygajlo, 2003] a larger dataset is used, which has an influence on the FAR and FRR. In addition, in this research only x and y coordinates are used for classification. In [Richiardi and Drygajlo, 2003] more features are added to gain a lower EER. Finally, the sampling rate at which the signatures are gathered is unknown in this experiment and that described in [Richiardi and Drygajlo, 2003]. More data can mean more information and higher accuracy, but too much data can mean lower accuracy. Unfortunately, it is not known to what extend the sampling rate affects the observed accuracy.

All in all, it can be concluded that, comparing these results with the EPWV technique results, GMM is less accurate when applied to Anoto technology. Furthermore, increasing the components has a slight advantage for certain values, but still the EER is higher than that of EPWV. Comparing the best EER of the GMM technique with the worst of EPWV, the difference is approximately 3 % in favor of EPWV.
These somewhat disappointing results, however do not exclude GMM entirely from being applied to Anoto technology. In fact, there is much room left for improvement and it would not be surprising if, when extra features and a larger signature dataset are used, GMM outperforms EPWV eventually. However, EPWV has also a lot of room for improvement. Therefore, it is difficult to predict which technique is eventually more accurate. One thing is evident: EPWV classification completes much faster than the GMM technique.

9 Future work

Although the results that are yielded in the experiments of this project classify most of the signatures in a correct way, for some environments these results may be too high. These environments could be, for example banks, for which signature verification techniques form a critical part. With additional adjustments however, the techniques are able to decrease the error rate. In this chapter a description will be given of which parts of the techniques have to be adjusted in order to produce a lower error rate. Furthermore, adjustments regarding the framework of the applications that have been developed are considered.

9.1 Technique

Compared to the GMM technique, EPWV yields a lower error rate. However both techniques have error rates which, unfortunately, are still too high for applications that rely heavily on signature verification. Various improvements can be added in the future to yield a lower and more reliable error rate.

1. Adding velocities, accelerations, trajectory angles and global features
2. Normalizing the size of signatures
3. Increasing the signature set
4. Using individual thresholds

The first improvement contains the addition of local features, such as the pen's velocity, acceleration and trajectory angles between coordinates. These improvements have already proven to yield a lower EER, such as in [Gupta and Joyce, 2007], [Rhee, Cho and Kim, 2001] and [Fierrez-Aguilar, et al., 2005].

Normalizing the size of signatures is the second suggested improvement and part of preprocessing. Consider for example two signatures that belong to the same person, but are of different sizes. Changing the height and width of all signatures to one fixed size, while maintaining the information about the coordinates, reduces the risk of rejecting a genuine signature. The Gaussian mixture model will benefit more from this technique than the extreme point warping variant technique, since the latter looks only at the minima and maxima, which will not change much when normalizing the sizes. The former technique maintains information about all coordinates and the displacements of these coordinates may differ when considering different signature sizes. Normalizing the size may yield in a lower false acceptance rate.

The third improvement suggests increasing the set of available signatures. This will lead to a more reliable EER. The main problem with using small datasets is that the EER will fluctuate more than when using large datasets. Furthermore, a larger dataset represents reality more than a small dataset. Finally, a large dataset allows more variations in the training, test and forgery sets. A good example of using different signature sets is discussed in [Nalwa, 1997].

The last improvement is using individual threshold as discussed in [Gupta and Joyce, 2007]. Up until now a global threshold is used, i.e.: for every person a fixed threshold is determined. In [Gupta and Joyce, 2007] it is shown that with an individual threshold a lower EER can be yielded. For the GMM technique the Gaussian mixture components are defined global as well. In [Richiardi and Drygajlo, 2003] it is shown that the introduction of user dependant models outperforms a fixed model system.

9.2 Framework

In the previous section it was discussed which adjustments can be added to lower the EER of the techniques used in this experiment. In this section, adjustments regarding the framework of the applications of the techniques are discussed. These adjustments are listed as follows:

1. Automating the process of converting signatures to a human readable format
2. Developing the framework with ASP.NET only
3. Using a MSSQL database for the signatures

As explained before, all signatures have to be converted to a readable format by hand. This process can be automated by accessing the source code of MyScript Notes and using the “save as” functionality to store the encrypted signatures as a readable decrypted format. Unfortunately the source code is not freely available. It has to be determined if the gain in costs of automating the process minus buying the license is worthwhile. On the long run it is expected to be, since no human intervention is needed for verifying a signature, while this process may be executed at large amounts per day based on the company for which it is used.

The second adjustment implies using one technology only. In this case it is possible to implement everything with ASP.NET. The advantage of this approach is that no connections, in which compatibility errors may occur, to other frameworks need to be maintained. It remains however unclear how the GMM technique (if this technique is chosen) can be translated to .NET environment, since currently this is not possible. Furthermore, .NET has the possibility to build a webpage and eventually host this page on a server in such a way that it can be accessed from anywhere. This is extremely useful in the context of the problem statement defined early in this thesis. To be precise, personnel that possess consignment notes are continually mobile. With the Bluetooth connection on the pen, the link to a mobile phone can be made easily, since this feature is included as a basic functionality of the digital pen. Next, assumed the mobile phone has internet access, the file can be submitted and in the ideal case, immediate feedback is given whether the signature is accepted or rejected. This immediate feedback can only be given if the first adjustment discussed before is added to the framework.

The last adjustment suggests using a relational database structure (in this case MSSQL) for the signatures. This means that signatures are put into a database in such a way that different operations on these signatures can easily be made. Currently, the signatures are stored as files. This is less efficient than a database, but for the prototype it suffices, since the amount of signatures is relatively small. When having thousands of signatures, however, a substantial performance boost may be experienced when using a relational database.

10 Conclusion

Different signature verification techniques have been studied and the Extreme Point Warping Variant (EPWV) and Gaussian Mixture Models (GMMs) were chosen to be integrated in Anoto environment. Looking back at the results it can be concluded that EPWV has a lower error rate than GMMs in the environment used in this project. It can also be concluded that, although the error rates are lower than those produced by GMMs, they may be still too high for a real world application. Fortunately there is enough space for improvement and I believe EPWV will show more accurate results and will prove itself powerful in a real world application when all the improvements discussed in the future work are added to the current model.
Next to lower error rates it is shown that EPWV can be integrated into Anoto environment seamlessly by using .NET technology. The execution time of EPWV is also considerably lower than of GMMs, where classification can take up to more than an hour.

All in all, it can be concluded that from the two chosen techniques, EPWV is the most promising and powerful solution which is able to verify signatures reliably within Anoto environment. Additionally, for real world environments, EPWV contains sufficient improvement space to adhere to these environments' requirements.
For Logica, this solution can mean an advantage to its competitors. Logica has a wide range of customers, including transport companies that could be interested in an automatic signature verification solution. Further development of the solution provided in this thesis can make the difference in convincing existing customers. Furthermore, potentially new customers may be acquired, when the solution appears to be successful.

Summary

In this document, important decisions are made on which the project relies during its lifecycle.

The document starts with explaining Anoto technology. After this explanation, decisions are made on which hardware is going to be used throughout the project's lifecycle. Alternatives are considered and motivations are given about using a certain combination of hardware. Next to the hardware, the software that has to be coupled to this hardware is defined. The combination of different applications and their independent roles that are used to realize the overall functionality of the system have to be selected. All in all, Anoto technology, hardware and software define the first part of this document.

The second part of the document consists of the underlying signature verification techniques that are applied in Anoto environment. Different techniques are considered and explained. Next, a small selection of two techniques that are applied is given. These two techniques are Gaussian Mixture Models (GMMs) and a variant of Extreme points warping (EPWV). These techniques are specified with a general approach and references are given for readers interested in the technical details of these techniques.

Of these two techniques, results are gathered and suggested improvements are given. Finally, a conclusion is given, in which statements are made about the reliability of the two mentioned techniques and Anoto technology. These statements are based on the experiment results and the research that led to these results.

References

AnotoHP. *Anoto.* http://www.anoto.com [accessed February 4, 2008].

AnotoPP. http://partner.anoto.com/.

Bajaj, R., and S. Chaudhury. "Signature verification using multiple neural classifiers." *Pattern Recognition* [Elsevier] 30, no. 1 [1997]: 1--7.

BioID. *About FAR, FRR and EER.* HumanScan GmbH. 2004. http://www.bioid.com/sdk/docs/About_EER.htm [accessed June 2008].

BioWeb. *BioWeb.* 1997. http://et.wcu.edu/aidc/BioWebPages/Biometrics_Signature.html [accessed 2008].

Degerholm, M.A. "Digital recognition and verification of handwritten signatures." 2005.

Dolfing, J.G.A., E.H.L. Aarts, and J. Oosterhout. "On-line signature verification with Hidden Markov Models." *Proceedings of the Fourteenth International Conference on Pattern Recognition* 2 [1998]: 1309--1312.

Feng, H., and C.C. Wah. "Online signature verification using a new extreme points warping technique." *Pattern Recognition Letters* [Elsevier] 24, no. 16 [2003]: 2943--2951.

Fierrez-Aguilar, J., L. Nanni, J. Lopez-Penalba, J. Ortega-Garcia, and D. Maltoni. "An on-line signature verification system based on fusion of local and global information." *Audio-and Video-based Biometric Person Authentication* [Springer], 2005: 20--22.

Fuentes, M., S. Garcia-Salicetti, and B. Dorizzi. "On line signature verification: Fusion of a Hidden Markov Model and a neural network via a support vector machine." *Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition*, 2002: 253--258.

Guo, J.K., D. Doermann, and A. Rosenfeld. "Local correspondence for detecting random forgeries." *Proceedings of the Fourth International Conference on Document Analysis and Recognition* 1 [1997].

Gupta, G.K., and R.C. Joyce. "Using position extrema points to capture shape in on-line handwritten signature verification." *Pattern Recognition* [Elsevier] 40, no. 10 [2007]: 2811--2817.

Haykin, Simon. "Neural Networks." In *Neural Networks*, by Simon Haykin. New York: Macmillian College Publishing Company, Inc., 1994.

Io2SW. *About digital writing.* Logitech. http://www.logitech.com/index.cfm/66/444&cl=us,en [accessed February 2008].

Jung, K.C., S.M. Yoon, and H.J. Kim. "Continuous HMM applied to quantization of on-line Korean character spaces." *Pattern Recognition Letters* [Elsevier] 21, no. 4 [2000]: 303--310.

Kashi, R., J. Hu, W.L. Nelson, and W. Turin. "A Hidden Markov Model approach to online handwritten signature verification." *International Journal on Document Analysis and Recognition* [Springer] 1, no. 2 [1998]: 102--109.

Ketabdar, H., J. Richiardi, and A. Drygajlo. "Global feature selection for on-line signature verification." *Proceedings International Graphonomics Society 2005 Conference*, June 2005.

Kholmatov, A. "Biometric Identity Verification Using On-Line & Off-Line Signature Verification." *Master's thesis, Sabanci University*, 2003.

Kholmatov, A., and B. Yanikoglu. "Identity authentication using improved online signature verification method." *Pattern Recognition Letters* [Elsevier] 26, no. 15 [2005]: 2400-2408.

Koh Chin Wei, E. "Maximum Likelihood Classification of Audio Segments with Gaussian Mixture Models." 2006.

Lee, L.L., T. Berger, and E. Aviczer. "Reliable On-Line Human Signature Verification Systems." [IEEE Computer Society] 1996.

Lin, C.F., and C.W. Chen. "A new approach to the verification of chinese signatures with variant orientations and scales using relaxation and state-space search methods." *Pattern Recognition* [Elsevier] 31, no. 6 [1998]: 665-674.

Lv, H., and W. Wang. "On-Line Signature Verification Based on Dynamic Bayesian Network." *Lecture notes in computer science* [Springer] 4221 [2006]: 507.

Martens, R., and L. Claesen. "Dynamic programming optimisation for on-line signature verification." *Proceedings of the 4th International Conference on Document Analysis and Recognition* [IEEE Computer Society Washington, DC, USA], 1997: 653-656.

Martens, R., and L. Claesen. "On-Line Signature Verification by Dynamic Time-Warping." *Proc. ICPR*, 1996: 38-42.

Matlab. *Matlab.* Mathworks. http://www.mathworks.com/products/matlab/.

Mistrut, D. *An implementation of the Wagner-Fischer edit distance.* 2002. http://search.cpan.org/~davidebe/Text-WagnerFischer-0.04/WagnerFischer.pm [accessed April 1, 2008].

Moore, A.W. *Clustering with Gaussian Mixtures.* 2005.

MSNotes. *MyScript Notes.* http://www.visionobjects.com/products/application-software/myscript-notes/.

Munich, M.E., and P. Perona. "Continuous dynamic time warping for translation-invariant curve alignment with applications to signature verification." *Proceedings of the Eighth IEEE International Conference on Computer Vision*, 1999.

Nabney, I., and C.M. Bishop. *Netlab.* 2001. http://www.ncrg.aston.ac.uk/netlab.

Nalwa, V.S. "Automatic on-line signature verification." *Proceedings of the IEEE* 85, no. 2 [1997]: 215--239.

Plamondon, R., and G. Lorette. "Automatic signature verification and writer identification—the state of the art." *Pattern Recognition* [Elsevier] 22, no. 2 [1989]: 107-131.

Plamondon, R., and S.N. Srihari. "On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey." [IEEE Computer Society] 2000.

Quek, C., and R.W. Zhou. "Antiforgery: a novel pseudo-outer product based fuzzy neural network driven signature verification system." *Pattern Recognition Letters* [Elsevier] 23, no. 14 [2002]: 1795-1816.

Rabiner, L.R. "A tutorial on hidden Markov models and selected applications in speech recognition." [Morgan Kaufmann Publishers Inc. San Francisco, CA, USA] 1990.

Reynolds, D.A., T.F. Quatieri, and R.B. Dunn. "Speaker Verification Using Adapted Gaussian Mixture Models." *Digital Signal Processing* [Elsevier] 10, no. 1-3 [2000]: 19-41.

Rhee, T., S. Cho, and J. Kim. "On-Line Signature Verification Using Model-Guided Segmentation and Discriminative Feature Selection for Skilled Forgeries." *Proceedings of the Sixth International Conference on Document Analysis and Recognition*, 2001: 645-649.

Richiardi, J., and A. Drygajlo. "Gaussian Mixture Models for on-line signature verification." *Proceedings of the 2003 ACM SIGMM workshop on Biometrics methods and applications* [ACM Press New York, NY, USA], 2003: 115-122.

Richiardi, J., H. Ketabdar, and A. Drygajlo. "Local and Global Feature Selection for On-line Signature Verification." *Proceedings of the Eighth International Conference on Document Analysis and Recognition*, 2005: 625-629.

Sanderson, C., and K.K. Paliwal. "Fast features for face authentication under illumination direction changes." *Pattern Recognition Letters* [Elsevier] 24, no. 14 [2003b]: 2409-2419.

Sanderson, C., and S. Bengio. "Robust Features for Frontal Face Authentication in Difficult Image Conditions." *IDIAP-RR 03* [Springer] 5 [2003a].

Stafford, S.J. "The Sequential GMM: A Gaussian Mixture Model Based Speaker Verification System that." 2005.

Stergiou, C., and D Siganos. *Neural Networks.* 1996. http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html [accessed March 2008].

VOHP. http://www.visionobjects.com/.

Wagner, R.A., and M.J. Fischer. "The String-to-String Correction Problem." *Journal of the ACM (JACM)* [ACM Press New York, NY, USA] 21, no. 1 [1974]: 168-173.

Warakagoda, N. *Hidden Markov Models.* May 10, 1996. http://jedlik.phy.bme.hu/~gerjanos/HMM/node2.html [accessed July 14, 2008].

Yang, L., B.K. Widjaja, and R. Prasad. "Application of hidden Markov models for signature verification." *Pattern Recognition* [Elsevier] 28, no. 2 [1995]: 161-170.

Appendix A: Comparison table for various signature verification techniques

Table VIII: Performance of on-line signature verification techniques

Authors	Input signals u(t) and feature description	Training and/or test data base specimens(S) x writers(W)	Comparison method	Error rates	Comments
Achemlal, Mourier, Lorette and Bonnefoy(1986)	$u_x(t), u_y(t)$ 10 personalized parameters selected among 40	600 genuines training set: (10S x 60W) 600 forgeries 1 imitator	Weighted city-block distance	FRR: 11% FAR: 8%	Use of skilled forgeries 3 trials
Beatson (1986)	$u_x(t), u_y(t)$ Movements of the pen in the air included, spatial and dynamic features	1000 genuines 100 forgeries	Unknown	FRR: 1% FAR: 2%	No information on training set. Commercialized by Signify Corp. as SIGNIFY system
Bechet (1984)	$u_{vx}(t), u_{vy}(t)$ normalization	300 genuines training set: (4 or 5 S x 48W) 1500 forgeries 12 imitators	Euclidean distance and score	FRR: 5% FAR: 5%	Over 3 months
Bonnefoy and Lorette (1981)	$u_x(t), u_y(t)$ parameters selected from a training set of 18	342 genuines (15S x 14W): training (12S x 11W): test No forgeries	Sequential recognition decision-tree	FRR: 0-6% FAR: -	Only one trial over one year
Brault and Plamondon (1984)	$u_a(t)$ average acceleration, sum of accelerations, number of samples	243 genuines (5S x 50W)	Histogram classifier global and local likeliness coefficients	FRR: 1.2% FAR: 1%	Among 243 signatures random forgeries
Chorley, Olding, Parks, Pobgee and Watson (1975)	$u_x(t), u_y(t)$ 10 parameters selected among 100	200 genuines (5S x 70W): training 549 forgeries 40 imitiations	Windows in 10 dimensions space	FRR: 4% FAR: 0-0.05%	After 4 weeks reference: 5 consistent signatures 3 trials random and skilled forgeries commercialized by Transaction Security as VERISIGN and SECURISIGN systems.
Crane and Ostrem (1983)	$u_{fx}(t), u_{fy}(t), u_p(t)$ 25 parameters selected among 44 reference vectors	5220 genuines (2 x 10S) x 58W: training set 648 forgeries 12 imitators	Weighted Euclidean distance (standard deviation distance)	FRR: 1.5% FAR: 1.5% FRR: 2.25% FAR: 3%	Over 4 months – 3 trials Random forgeries Skilled forgeries

Authors	Input signals u(t) and feature description	Training and/or test data base specimens(S) x writers(W)	Comparison method	Error rates	Comments
DeBruyne (1985)	$u_x(t), u_y(t)$ 4 parameters selected among 18	71 genuines from 11W 52 forgeries	Weighted parameters grading maximal-likelihood ratio test	FRR: 3% FAR: 2%	Reference parameters evaluated from 10 specimens
Dyche (1969)	$u_x(t), u_y(t)$ 40 parameters based on time length position, speed and acceleration self and cross moments	Training: 333 genuines from 1 signer 167 forgeries from 3 amateur forgers Tests: 125 signatures from each class	Likelihood ratio nearest neighbor linear boundary	FRR: 0% FAR: 0%	Similar results using a subset of 15 parameters
Farag and Chien (1972)	$u_x(t), u_y(t)$ Recording duration 2S (256 coordinates) chain coded vector histogram	200 genuines (20S x 10W) 250 forgeries: 10 users imitated 5 imitators	City block distance Mahalanobis distance	FRR: 14-23% FAR: 14-23%	None
Groupement, Carte Bleue (1984)	$u_x(t), u_y(t)$ Signature coded into 40 bytes of data	5656 genuines (5S x 460W): training 3509S: test	Correlation 0-99	FRR: 3.5% FAR: -	Reference 3 consistent signatures 3 trials over one year credit card data in a public store Commercialized by Quest Micropad, as Q-SIGN system
Haberman and Fejfar (1976)	$u_p(t)$ 9 local or global features	209W (170 Male and 39 Female signatures) 2645 FRR tests 106505 FAR tests	Unknown	FRR: 6.81% FAR: 3.19%	Higher error rates for females using the VERIPEN system
Hale and Paganini (1980)	$u_f(t)$ 15 Haar coefficients 18 waveform physical features	Typical experiments: 500 genuines 97S 951 verification attempts 181 forgery attempts (non-observing signers) 59 forgery attempts (observing signers)	Weighted distance	FRR: 1.5% FAR: 1.2% (non-observing signers) 2.5% (observing signers)	None

Authors	Input signals u(t) and feature description	Training and/or test data base specimens(S) x writers(W)	Comparison method	Error rates	Comments
Herbst and Liu (1979)	$u_{ax}(t), u_{ay}(t), u_p(t)$	6000 genuines (6S x 201W) : training Forgeries: 40 users imitated 10S per user 40 imitators	Regional correlation	FRR: 1.7% FAR: 0.4% FRR: 0.022%	3 trials – over 6 months Skilled forgeries With random forgeries
Herbst and Liu (1979)	$u_{ax}(t), u_{ay}(t)$	1042 gebuines (5S x 40W): training 750 forgeries	Unknown	FRR: 2.4% FAR: 3.2%	3 trials
Ibrahim and Levrat (1979)	$u_x(t), u_y(t), u_v(t)$ $u_a(t)$ Global parameters Peak matching Zero crossing	47 genuines (1S x 10W): training 36 forgeries 2 imitators	Euclidean distance	FRR: 19% FAR: 5.5%	Signing with a finger instead of a pen
Lorette (1983)	$u_x(t), u_y(t)$ 7 Global and local dynamic parameters	210 genuines (5S x 14W): training (10S x 14W): test No forgeries	Clustering analysis	FRR: 6% FAR: unknown	Only one trial
Mauceri (1965)	$u_a(t), u_p(t)$ (pen paper contact) 10 indices related to frequency spectra 19 local and global features	2350 genuines 45S x 40W No forgeries 20S/W learning 25S/W test	Squared distance Weighted distance	60% recognition rate 90% recognition rate	Partial results from 10 subjects only
Sato and Kogure (1982)	$u_x(t), u_y(t), u_p(t)$ Complex normalized functions shape, motion and pressure vector	110 genuines (10S x 11W) 330 forgeries 1 user 10S/user 3 imitators	Dynamic programming matching Mahalnobis distance Pseudo-distance	FRR: 1.8% FAR: close to 0%	No information on the training set
Sternberg (1975)	$u_p(t)$ Resuced to a few “measures” via mathematical transformation	1000 genuines (10S x 100W) 50 imitators Each one forging 8 subjects’ signatures 3 times Random forgeries		FRR: 0.7% FAR: 1.8%	Over 2 months with deliberate forgery Commercialized by veripen as SIGNAC system

Authors	Input signals u(t) and feature description	Training and/or test data base specimens(S) x writers(W)	Comparison method	Error rates	Comments
Worthington, Chainer, Williford and Gundersen (1985)	$u_{ax}(t), u_{ay}(t), u_p(t)$	5000 genuines (6S x 108W): training 4700S: test 2133 forgeries	5 similarity measures	FRR: 1.77% FAR: 0.28% 2.33%	Tests over 9 months
Yasubata and Oka (1977)	$u_x(t), u_y(t), u_p(t)$ Calculated force function	100 genuines (3S x 10W): training (7S x 10W): no training	D.P. matching Euclidean distance	FRR: 1.3% FAR: <1%	Key words used instead of signature
Zimmerman and Varady (1985)	$u_p(t)$ (one bit) Walsh functions Power spectrum 40 low sequency harmonics	90 genuines (3S x 9W): training	Fisher discriminate functions Linear classifier	FRR: 30-50% FAR: 4-12%	Over several weeks
Zimmerman and Werner (1978)	$u_x(t), u_y(t)$ Acceleration derived from FIR filter	Not specified	Straightforward digital correlation	Less than 10% misrecognition	Preliminary tests

Appendix B: Digital pen comparison table

Table IX: Digital pen comparison

	Nokia SU-27W	Maxell Penit (DP-201)	Logitech Io2	Sony Ericsson Chatpen CHA-30
Weight	37.5 g	30g	37.4 g	45 g
Dimensions	157 x 24 x 21 mm	157 x 21 x 18mm	156.7 mm (length)	165 x 25 x 20mm
Memory of data	~1.3 MB (more than 100 full A5 pages)	1 MB (approximate 40 A5 pages)	856 KBytes (approximate 40 A5 pages)	Corresponding to 40 A4 Written Pages
Bluetooth	YES (v1.2)	YES (v1.2)	YES	YES (v1.1)
USB connection	YES (connectivity stand)	YES (USB v1.1, cradle)	YES (cradle)	NO
Operation time	Up to 3 hours	Minimum of 2 hours	Up to 3 hours	Up to 2 hours
Standby time	Up to 20 hours	Minimum of 10 hours	Up to 20 hours	10 hours
Operation temperature	+0°C to +55°C	+0°C to +40°C	Unknown	Unknown
Battery	Rechargeable Li-Ion 180mAh	Lithium-ion rechargeable battery	Rechargeable lithium ion battery	Rechargeable Li-Polymer
Package contents	Nokia Digital Pen SU-27W	Maxell DP-201 Digital Pen with 3 refills	Logitech® io™2 Digital Pen	Sony Ericsson Chatpen CHA-30
	Connectivity Stand DT-17	PC-201 Travel Cradle including spare USB cable	Smart paper: A5 notebook	Instructions manual
	Travel Charger AC-3	CB-201 Cradle Desk Stand	Travel cradle with USB cable	Post-it note pad

(Continued)	Nokia SU-27W	Maxell Penit (DP-201)	Logitech Io2	Sony Ericsson Chatpen CHA-30
Package contents	Nokia Charging Adapter AD-48	Oxford Easybook® M^3 A5 Digital Notebook (64 sheets)	3 ink refills	Note pad (43 A5 pages)
	Digital paper: MMS pad (B7 size) and Notes pad (A5 size)	BlackBerry sticker for Easybook® M^3 Digital Notebook	Installation and reference CD, including Logitech® io™2 software and Microsoft® .NET Framework 1.1	· Charger, Spare pen refills
	Pin-code card	Digital Pen for BlackBerry User Guide	2-year limited warranty	
	CD-rom	12 months subscription to PaperIQ handwriting recognition & fax service		
	User guide			
Handwriting recognition included?	YES (application software on CD)	PARTLY (via the one-year subscription)	YES (same as Nokia)	Unknown
Price	€ 136.19	£ 189.95 (€ 254.70)	$199.99 (€ 137.81)	$ 99.00 (€ 64.10)
Order time	Short (within a week)	Unknown	Unknown (product is partly taken out of market)	

Appendix C: String distance algorithm

```
private void setDistance(String first, String[] second)
    {
        int first_word_length = first.Length;
        result = new int[second.Length];

        for (int current_word = 0; current_word < second.Length; current_word++)
        {
            int second_word_length = second[current_word].Length;
            int[,] cost_matrix = new int[first_word_length + 1, second_word_length + 1];


            if (first_word_length == 0)
            {
                result[current_word] = second_word_length * costs[1];
            }
            if (second_word_length == 0)
            {
                result[current_word] = first_word_length * costs[1];
            }

            cost_matrix[0, 0] = 0;

            // Fill the first column with 0, 1, 2, 3, etc
            for (int i = 1; i <= first_word_length; i++)
            {
                cost_matrix[i, 0] = i * costs[1];
            }
            // Fill the first row with 0, 1, 2, 3, etc
            for (int j = 1; j <= second_word_length; j++)
            {
                cost_matrix[0, j] = j * costs[1];
            }

            /*
             * The core of the algorithm that computes the distances
             * Note that the optimal distance is put in cost_matrix[x, x],
             * i.e.: on the diagonal.
             */
            for (int i = 1; i <= first_word_length; i++)
            {
                String first_word_char = first.Substring(i - 1, 1);

                for (int j = 1; j <= second_word_length; j++)
                {
                    String second_word_char = second[current_word].Substring(j - 1, 1);

                    cost_matrix[i, j] = min(
                      cost_matrix[i - 1, j] + weight(first_word_char, "-"),
                      cost_matrix[i, j - 1] + weight("-", second_word_char),
                      cost_matrix[i - 1, j - 1] + weight(first_word_char, second_word_char)
                      );
                }
            }
            result[current_word] = cost_matrix[first_word_length, second_word_length];
        }
    }
```

Appendix D: EPWV result tables

All features							
Signatures accepted	**Signatures rejected**	**Signatures accepted falsely**	**Signatures rejected falsely**	**Threshold**	**FAR**	**FRR**	**EER**
86	274	0	94	0	0	0,52	-
122	238	0	58	0,3	0	0,32	-
135	225	0	45	0,4	0	0,25	-
112	248	0	68	0,5	0	0,38	-
142	218	0	38	1	0	0,21	-
171	189	13	22	1,5	0,07	0,12	-
179	181	16	17	1,765	0,09	0,09	**0,09**
184	176	14	10	2	0,08	0,06	-
187	173	15	8	2,5	0,08	0,04	-
195	165	16	1	3	0,09	0,01	-
260	100	80	0	6	0,44	0	-
301	59	121	0	10	0,67	0	-

Table X: All features

Only magnitudes							
Signatures accepted	Signatures rejected	Signatures accepted falsely	Signatures rejected falsely	Threshold	FAR	FRR	EER
88	272	0	92	0	0	0,51	-
119	241	0	61	0,3	0	0,34	-
124	236	3	59	0,4	0,02	0,33	-
133	227	6	53	0,5	0,03	0,29	-
152	208	6	34	1	0,03	0,19	-
182	178	20	18	1,45	0,11	0,1	**0,1**
182	178	19	17	1,5	0,11	0,09	-
217	143	43	6	2,5	0,24	0,03	-
226	134	48	2	3	0,27	0,01	-
301	59	121	0	6	0,67	0	-
340	20	160	0	10	0,89	0	-

Table XI: Magnitudes only

Only Time info							
Signatures accepted	Signatures rejected	Signatures accepted falsely	Signatures rejected falsely	Threshold	FAR	FRR	EER
98	262	0	82	0	**0**	**0,46**	-
118	242	0	62	0,3	0	0,34	-
124	236	0	56	0,4	0	0,31	-
129	231	0	51	0,5	0	0,28	-
145	215	0	35	1	0	0,19	-
173	187	12	19	1,5	0,07	0,11	-
180	180	16	16	1,85	0,09	0,09	**0,09**
186	174	17	11	2	0,09	0,06	-
191	169	14	3	2,5	0,08	0,02	-
192	168	15	3	3	0,08	0,02	-
253	107	73	0	6	0,41	0	-
300	60	120	0	10	0,67	0	-

Table XII: Time info only

None of the above							
Signatures accepted	**Signatures rejected**	**Signatures accepted falsely**	**Signatures rejected falsely**	**Threshold**	**FAR**	**FRR**	**EER**
92	268	0	88	0	0	0,49	-
114	246	8	74	0,3	**0,04**	**0,41**	-
122	238	4	62	0,4	0,02	0,34	-
136	224	7	51	0,5	0,04	0,28	-
136	224	9	53	1	0,05	0,29	-
179	181	18	19	1,46	0,1	0,11	**0,1**
187	173	23	16	1,5	0,13	0,09	-
208	152	33	5	2	0,18	0,03	-
220	140	46	6	2,5	0,26	0,03	-
237	123	60	3	3	0,33	0,02	-
306	54	126	0	6	0,7	0	-
335	25	155	0	10	0,86	0	-

Table XIII: Only basic extreme points information

Appendix E: GMM result tables

Components	Covariance type	K-means	EM	Threshold	FAR	FRR	EER
8	Diagonal	5	5	1	0,1744	0,2255	-
8	Diagonal	5	5	1,2	0,1613	0,1684	**0,16**
8	Diagonal	5	5	1,5	0,19	0,1477	-
8	Diagonal	5	5	2	0,2564	0,0986	-
8	Diagonal	5	5	2,5	0,2787	0,0909	-
8	Diagonal	5	5	3	0,2937	0,0806	-
8	Diagonal	5	5	6	0,38	0,0263	-
8	Diagonal	5	5	10	0,4689	0	-

Table XIV: GMM results with 8 components

Components	Covariance type	K-means	EM	Threshold	FAR	FRR	EER
16	Diagonal	5	5	1	0,1143	0,2712	-
16	Diagonal	5	5	1,2	0,1324	0,2917	-
16	Diagonal	5	5	1,5	0,1266	0,2294	-
16	Diagonal	5	5	2	0,1444	0,1735	-
16	Diagonal	5	5	2,35	0,1579	0,1505	**0,15**
16	Diagonal	5	5	2,5	0,1531	0,1222	-
16	Diagonal	5	5	3	0,2018	0,0886	-
16	Diagonal	5	5	6	0,3404	0,0213	-
16	Diagonal	5	5	10	0,4503	0	-

Table XV: GMM results with 16 components

Components	Covariance type	K-means	EM	Threshold	FAR	FRR	EER
24	Diagonal	5	5	1	0,0926	0,3358	-
24	Diagonal	5	5	1,2	0,0714	0,3182	-
24	Diagonal	5	5	1,5	0,0794	0,288	-
24	Diagonal	5	5	2	0,0946	0,2368	-
24	Diagonal	5	5	2,5	0,1279	0,1863	-
24	Diagonal	5	5	3	0,1222	0,1531	-
24	Diagonal	5	5	3,15	0,1277	0,1277	**0,1277**
24	Diagonal	5	5	6	0,3077	0,069	-
24	Diagonal	5	5	10	0,4303	0	-

Table XVI: GMM results with 24 components

Components	Covariance type	K-means	EM	Threshold	FAR	FRR	EER
32	Diagonal	5	5	1	0,1026	0,396	-
32	Diagonal	5	5	1,2	0,1053	0,4	-
32	Diagonal	5	5	1,5	0,087	0,3662	-
32	Diagonal	5	5	2	0,1644	0,287	-
32	Diagonal	5	5	2,5	0,1585	0,2358	-
32	Diagonal	5	5	3	0,125	0,2222	-
32	Diagonal	5	5	3,39	0,1684	0,1613	**0,16**
32	Diagonal	5	5	6	0,2913	0,0656	-
32	Diagonal	5	5	10	0,4125	0	-

Table XVII: GMM results with 32 components

Appendix F: List of abbreviations

Abbreviation	Description
ASP	Active Server Pages
ANN	Artificial Neural Network
CDTW	Continuous Dynamic Time Warping
DTW	Dynamic Time Warping
EER	Equal Error Rate
EM	Expectation Maximization
EPW	Extreme Points Warping
EPWV	Extreme Points Warping Variant
FAR	False Acceptance Rate
FDK	Form Design Kit
FRR	False Rejection Rate
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
SDK	Software Development Kit
USB	Universal Serial Bus

Appendix G: List of Figures, tables and equations

Figures

Tables

Equations