MODELS AND ALGORITHMS IN OFF-LINE, FEATURE-BASED, HANDWRITTEN SIGNATURE VERIFICATION

PhD Thesis

Bence Andras Kovari

Advisor:
Dr. Hassan Charaf Ph.D.
Associate Professor

Budapest University of Technology and Economics
Department of Automation and Applied Informatics

Budapest, 2013
# Table of contents

Table of contents ................................................................................................................ 2  
List of tables and figures ............................................................................................... 5  
Abstract .......................................................................................................................... 7  
Összefoglaló .................................................................................................................... 8  
Preface ............................................................................................................................ 9  
Declarations ..................................................................................................................... 9  
Nyilatkozat .................................................................................................................... 9  
Acknowledgements ......................................................................................................... 10  
Köszönetnyilvánítás ........................................................................................................ 10  
1. Introduction ................................................................................................................ 11  
  1.1 Thesis contributions ................................................................................................. 12  
  1.2 Thesis structure ...................................................................................................... 13  
2. Motivation and background ...................................................................................... 14  
  2.1 Terms and concepts ................................................................................................. 14  
  2.2 Performance metrics ............................................................................................... 15  
  2.3 The human factor .................................................................................................... 16  
  2.4 Artificial forgeries .................................................................................................. 17  
  2.5 State of the art .......................................................................................................... 17  
  2.6 The verification process ........................................................................................... 18  
    2.6.1 Acquisition ......................................................................................................... 19  
    2.6.2 Preprocessing .................................................................................................... 19  
    2.6.3 Feature extraction ............................................................................................... 19  
    2.6.4 Processing ........................................................................................................... 20  
    2.6.5 Classification ..................................................................................................... 20  
  2.7 Off-line signature verification systems ..................................................................... 21  
  2.8 Local features in signature verification ................................................................... 23  
  2.9 The feature matching problem ................................................................................. 24  
  2.10 Test databases, preprocessing, experimental setup ............................................... 25  
    2.10.1 BME-AUT1 ....................................................................................................... 25  
    2.10.2 BME-AUT2 ....................................................................................................... 25  
    2.10.3 BME-AUT9 ....................................................................................................... 26  
    2.10.4 SVC2004 .......................................................................................................... 26  
    2.10.5 GPDS ................................................................................................................. 26  
    2.10.6 MCYT ............................................................................................................... 27  
    2.10.7 CEDAR ............................................................................................................. 27
### Table of Contents

3. The statistical model of signature verification ........................................28  
   3.1 Related work ................................................................................28  
   3.2 The terminology of features ..........................................................29  
   3.3 The distribution of feature properties in original signatures ..........30  
   3.4 Experiments ..................................................................................31  
   3.5 The distribution of feature properties in forged signatures ..........33  
   3.6 Experiments ..................................................................................33  
   3.7 Distinguishing original and forged feature properties ..........35  
   3.8 Building a classifier ......................................................................38  
      3.8.1 Single feature property ..........................................................38  
      3.8.2 Multiple feature properties ...................................................39  
      3.8.3 Experimental results ............................................................42  
   3.9 Chapter summary ..........................................................................45  
4. The implications of the statistical model ........................................47  
   4.1 The effect of feature count \((k)\) on AER ..............................47  
      4.1.1 Case 1 ...................................................................................48  
      4.1.2 Case 2 ...................................................................................49  
   4.2 The effect of sample count \((n)\) on AER .............................51  
   4.3 The effect of threshold \((w)\) on AER .......................................53  
   4.4 Chapter summary .........................................................................57  
5. Signature processing algorithms ..................................................58  
   5.1 Line thinning ...............................................................................58  
      5.1.1 The classification of thinning algorithms .........................58  
      5.1.2 The Scanning Circle Algorithm ..........................................61  
      5.1.3 Comparison with other algorithms ....................................69  
      5.1.4 The experimental setup .......................................................70  
      5.1.5 Additional benefits of the SCA ...........................................72  
   5.2 Feature matching .................................................................72  
      5.2.1 Related work ........................................................................72  
      5.2.2 Matching reference signatures .............................................73  
      5.2.3 Matching sample signatures ................................................76  
   5.3 Feature models and extraction algorithms ............................77  
      5.3.1 Global features .....................................................................77  
      5.3.2 Baseline model .....................................................................77  
      5.3.3 Loop model ..........................................................................79  
   5.4 Chapter summary .........................................................................81  
6. The application of the results ......................................................83
6.1 Signature verification framework ................................................. 83
  6.1.1 Architecture ........................................................................ 83
  6.1.2 A modular system ................................................................. 84
  6.1.3 Education ............................................................................ 85
6.2 Analytical tool ........................................................................... 86
6.3 Silverlight Frontend ................................................................. 87
6.4 WPF Frontend ........................................................................... 88
6.5 Verification reports ................................................................. 88
7. Conclusions .................................................................................. 90
  7.1 Summary .................................................................................. 90
  7.2 Future work ............................................................................. 92
Appendix A Detailed proofs................................................................. 94
  A.1 Detailed proof of Proposition 3.2 ............................................. 94
  A.2 Detailed proof of Lemma 3.1 ................................................... 95
  A.3 Detailed proof of Proposition 3.3 ............................................. 96
  A.4 The calculation of $\alpha$ and $\beta$ based on Proposition 3.3 ....... 98
  A.5 Detailed proof of Proposition 3.4 ............................................ 100
  A.6 Details for the proof of Proposition 4.1 .................................... 101
    A.6.1 Details of equation 4.16: .................................................. 101
    A.6.2 Details for equation 4.23: .................................................. 104
    A.6.3 Details of equation 4.27: .................................................. 104
  A.7 Scripts for Proposition 4.2 ....................................................... 105
Appendix B The signature verification API........................................... 106
Appendix C SVC 2004 database samples .......................................... 108
8. Bibliography .................................................................................. 109
List of tables and figures

Figure 1.1 On-line verification requires special acquisition devices ............11
Figure 2.1 A typical off-line verifier with a threshold classifier Source: [26]...19
Figure 2.2 Using multiple features for the decision. Sources: [36] [37]........20
Figure 2.3 Separation of training and testing. Sources: [32] [33].............21
Figure 2.4 Combining global and local features. Source: [40]...................21
Figure 2.5 Feature-based off-line signature verification process .............24
Figure 2.6 Incorrect (left) and correct (right) matching of loops ..........24
Figure 2.7 Samples from the GPDS database .....................................26
Figure 3.1 Feature hierarchy of signatures (example).............................30
Figure 3.2. The Shapiro-Wilk test for baselines ...................................31
Figure 3.3. The Lilliefors test for baselines .......................................31
Figure 3.4. The Shapiro-Wilk test for loops .......................................32
Figure 3.5. The Lilliefors test for loops ...........................................32
Figure 3.6 The illustration of error rates ...........................................35
Figure 3.7 Optimal choices of decision threshold (w) .........................38
Figure 3.8 Probabilities of improper classification ..............................40
Figure 3.9 Predicted error rates ......................................................42
Figure 4.1 The relation of AER to the value of k at a given forgery quality (q)51
Figure 4.2 The function $\frac{\partial AER(q, n, k)}{\partial n}$ ..................................52
Figure 4.3 Relation of AER to the value of n at given forgery quality (q).....53
Figure 4.4 The relationship of q, w and AER when n = 10 and k = 10 ......54
Figure 4.5 Relation of AER to the value of w, at given forgery qualities (q)...54
Figure 4.6 The resulting errors ......................................................55
Figure 4.7 The resulting increase in error depending on the maximal value $q_{max}$ and the estimated value $q_{est}$ of forgery quality.........................56
Figure 4.8 The optimal choices for q, depending on the maximum of $q_{act}$ ...56
Figure 4.9 The maximal increase in error rate when choosing the optimal value of q for different maximal values.................................57
Figure 5.1 Endpoint extraction on a thinned image ................................58
Figure 5.2 Erosion of the blue square by a disk, resulting in the red square....59
Figure 5.3 Example of a commonly used thinning mask .......................60
Figure 5.4 Step 2 of the stroke extraction algorithm .............................62
Figure 5.5: Flowchart of the stroke extraction algorithm.......................63
Figure 5.6: Basic stroke extraction algorithm: stroke starts and ends are numbered and marked by the green and gray lines.......................64
**LIST OF TABLES AND FIGURES**

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.7</td>
<td>The stroke extraction algorithm detects 4 different strokes</td>
<td>65</td>
</tr>
<tr>
<td>5.8</td>
<td>A sample run of the improved stroke extraction algorithm</td>
<td>66</td>
</tr>
<tr>
<td>5.9</td>
<td>Stroke extraction</td>
<td>66</td>
</tr>
<tr>
<td>5.10</td>
<td>Improved stroke extraction</td>
<td>66</td>
</tr>
<tr>
<td>5.11</td>
<td>4 strokes are detected instead of the original 3</td>
<td>67</td>
</tr>
<tr>
<td>5.12</td>
<td>Average and standard deviation of stroke counts</td>
<td>68</td>
</tr>
<tr>
<td>5.13</td>
<td>Histogram of stroke count deviations in the original</td>
<td>68</td>
</tr>
<tr>
<td>5.14</td>
<td>Reference skeleton and the reconstructed signature</td>
<td>69</td>
</tr>
<tr>
<td>5.15</td>
<td>Reference skeleton (left) and thinning results (right)</td>
<td>70</td>
</tr>
<tr>
<td>5.16</td>
<td>Some examples of the thinning results</td>
<td>70</td>
</tr>
<tr>
<td>5.17</td>
<td>a) reference skeleton b) signature c) thinning results</td>
<td>71</td>
</tr>
<tr>
<td>5.18</td>
<td>Loop matching</td>
<td>73</td>
</tr>
<tr>
<td>5.19</td>
<td>The pairwise matching of signatures:</td>
<td>75</td>
</tr>
<tr>
<td>5.20</td>
<td>The combined graph of feature pairs</td>
<td>75</td>
</tr>
<tr>
<td>5.21</td>
<td>Iteration 1 of step 3</td>
<td>75</td>
</tr>
<tr>
<td>5.22</td>
<td>Iteration 2 of step 3</td>
<td>76</td>
</tr>
<tr>
<td>5.23</td>
<td>Iteration 3 of step 3, note that $d4$ is a false match</td>
<td>76</td>
</tr>
<tr>
<td>5.24</td>
<td>Matching a sample signature to four references</td>
<td>77</td>
</tr>
<tr>
<td>5.25</td>
<td>Baselines obtained by our algorithm</td>
<td>78</td>
</tr>
<tr>
<td>5.26</td>
<td>Phases of the baseline extraction process</td>
<td>79</td>
</tr>
<tr>
<td>5.27</td>
<td>Loops within a signature</td>
<td>79</td>
</tr>
<tr>
<td>5.28</td>
<td>Phases of the loop extraction process</td>
<td>81</td>
</tr>
<tr>
<td>6.1</td>
<td>Module selection and bulk processing</td>
<td>85</td>
</tr>
<tr>
<td>6.2</td>
<td>Manual feature matching</td>
<td>86</td>
</tr>
<tr>
<td>6.3</td>
<td>Debugging view of the loop extraction algorithm</td>
<td>86</td>
</tr>
<tr>
<td>6.4</td>
<td>Screenshots from the SigMan application</td>
<td>87</td>
</tr>
<tr>
<td>6.5</td>
<td>Interactive web interface (<a href="http://www.aut.bme.hu/signature">www.aut.bme.hu/signature</a>)</td>
<td>88</td>
</tr>
<tr>
<td>6.6</td>
<td>Screenshots from the “Signature Verifier” application</td>
<td>88</td>
</tr>
<tr>
<td>6.7</td>
<td>Parts of the verification report</td>
<td>89</td>
</tr>
<tr>
<td>2.1</td>
<td>Comparison table of off-line signature verifiers</td>
<td>22</td>
</tr>
<tr>
<td>3.1</td>
<td>The relation of original and forged baseline properties</td>
<td>34</td>
</tr>
<tr>
<td>3.2</td>
<td>The relation of original and forged loop properties</td>
<td>34</td>
</tr>
<tr>
<td>3.3</td>
<td>The relation of original and forged global properties</td>
<td>34</td>
</tr>
<tr>
<td>3.4</td>
<td>Classification results on the SVC2004 database</td>
<td>44</td>
</tr>
<tr>
<td>3.5</td>
<td>Classification results on the GPDS300 database</td>
<td>45</td>
</tr>
<tr>
<td>5.1</td>
<td>The average ratings, tested with 20 signatures</td>
<td>72</td>
</tr>
</tbody>
</table>
Abstract

The verification of handwritten signatures is one of the oldest and the most popular biometric authentication methods in our society. A history which spans several hundred years has ensured that it also has a wide legal acceptance all around the world.

As technology improved, the different ways of comparing and analyzing signatures became more and more sophisticated. Since the early seventies, people have been exploring how computers may aid and – maybe one day – fully take over the task of signature verification. Based on the acquisition process, the field is divided into on-line and off-line parts. In on-line signature verification, the whole process of signing is captured using some kind of an acquisition device, while the off-line approach relies merely on the scanned images of signatures. This thesis addresses some of the many open questions in the off-line field.

As the best current systems use artificial intelligence-based approaches, the insight we get into the reasons behind classification decisions is very limited. To achieve a better understanding of the whole process, my first contribution proposes a simplified probabilistic model for off-line signature verification. In this model, each of the verification steps can be mathematically described and, therefore, individually analyzed and improved. Using this model, it is possible to predict the accuracy of a signature verification system based on just a few a priori known parameters, such as the cardinality and the quality of input samples. Several experiments have been conducted using a statistics-based classifier to confirm the assumptions and the results of our model.

As next, some of the consequences of the previously introduced model are analyzed. It is proven that increasing the number of reference samples or the number of observed features will improve the achievable classification results. It has also been demonstrated that the only free parameter of our equation – which describes the quality of the forged signatures – can be specifically set to minimize the negative effects of a possible misestimation.

The next part of my thesis deals with feature extraction and matching algorithms used during the process of signature verification. A heuristic stroke extraction, a baseline extraction, a loop extraction and a feature-matching algorithm are introduced and described in detail.

In order to demonstrate the practical applications of the results, a complete signature verification framework has been developed, which incorporates all the previously introduced algorithms.

The results provided in this thesis aim to present a deeper analytical insight into the behavior of the verification system than the traditional artificial intelligence-based approaches.
Összefoglaló

A kézeredet azonosítás a legrégebbi és legnépszerűbb biometrikus azonosítási feladatok egyike, mely egyben több évszázados múltjának kőszönhetően világszerte széleskörű jogi elfogadottsággal is rendelkezik.

A technológia fejlődésével az aláírások ellenőrzésére is egyre kifinomultabb módszerek születtek, míg nem a 70-es évek kezdetétől a számítógépek is bevonták a folyamatba. A kezdeti kisegítő funkciókat követően, ma már az ellenőrzés teljes körű automatizálásának a lehetősége áll a kutatások középpontjában. Ezek a kutatások két nagy irányvonalra bonthatóak. Az on-line hitelesítés során az aláírás teljes folyamata rögzítésre kerül, míg az off-line módszerek kizárólag a már elkészült aláírások szkennelt képeinek az elemzésére korlátozódnak. Dolgozatomban ez utóbbi terület néhány nyitott kérdésére keresem a válaszokat.

Napjaink legsikeresebb aláírás-hitelesítő rendszerei többnyire mesterséges intelligencia (MI) alapú módszerekkel működnek, így a döntések miértbe sajnos kevés betekintést kapunk. Hogy mégis jobban megismerhessük a folyamatot, kidolgoztam egy statisztikai modellt a jellemző alapú off-line aláírás-hitelesítéshez. A modell elkülöníti az ellenőrzés fontosabb lépéseit, s lehetővé teszi, hogy azokat önmagukban elemezhessük, fejleszthessük, vagy akár matematikai módszerekkel leírjuk. A modell ezen felül lehetővé teszi, hogy néhány előre ismert paraméter (mintaszám, jellemzők száma, hamisítvány minősége) alapján előre megbecsüljük egy hitelesítő rendszer várható pontosságát. Modell emellett a folyamat részletes kézbesítését számos méréssel támasztom alá.


Harmadik tézisemben a korábbi eredményekhez szükséges feldolgozási lépéseket tárgyalom. Egy újítója heurisztikus vonalkövetési algoritmus mellett eljárásokat adtam az alapvonalak és hurkok kinyerésére, jellemzésére és párosítására is.

A gyakorlati alkalmazhatóság demonstrálásra megvalósítottam egy komplet aláírás-hitelesítési keretrendszer, mely – számos egyéb funkció felül megállapítja a fenti algoritmusokat is.

Eredményeim lehetővé teszik, hogy a hagyományos MI alapú módszerek nél mélyebb betekintést nyerhessünk az ellenőrzési folyamat részleteibe.
Preface

Declaration

The content of this thesis is a product of the author’s original work except where explicitly stated otherwise.

Nyilatkozat

Alulírott Kővári Bence András kijelentem, hogy ezt a doktori értekezést magam készítettem, és abban csak a megadott forrásokat használtam fel. Minden olyan részt, amelyet szó szerint, vagy azonos tartalomban, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Budapest, 2013. február 19.

________________________________________

(Kővári Bence András)

1 A bírálatok és a védésről készült jegyzőkönyv a későbbiekben a Dékáni Hivatalban elérhetőek.
Acknowledgements

This thesis could not have been created without the support of many people. First of all I am really grateful to my family, who have been standing by me all my life. Thanks to my friends, who reminded me again and again in different ways about the importance of my studies. I am indebted to my advisor, Hassan Charaf, for facilitating the human and the financial conditions of the work, and no less indebted to István Albert for similar reasons.

I am indebted to László Lengyel who helped me to find the way in the maze of scientific publications and devoted a significant amount of time to the proofreading of this work.

I am grateful to all of my students and coauthors who participated in my research in the past seven years. Their devotion and enthusiasm helped me to overcome the encountered dead ends and throw-backs, which unfortunately are the natural companions of basic research.

I would also like to express my gratitude to Mihály Bokor, Sándor Kolumbán Tamás Lukovszki whose advice inspired many improvements in my proofs.

I am indebted to Tamás Agárdi, the president of the Hungarian Institute of Graphology, who provided me with useful advice and background material for my research.

And last, but not least, I would like to thank Melinda Tünde Dóra, for correcting my spelling and grammar mistakes with infinite patience.

Köszönetnyilvánítás

Számos embernek tartozom hálával e dolgozat létrejöttéért. Mindenekelőtt a családomnak, akik egész életemben mellettem álltak és támogattak. Köszönöm a barátaimnak segítségüket, s hogy számos módon emlékeztették tanulmányaim fontosságára. Hálás vagyok témavezetőmnek Charaf Hassannak, aki megteremtette az anyagi és emberi feltételeket a kutatásomhoz és Albert Istvánnak, aki ebben szintén fontos szerepet játszott.

Köszönöm Lengyel Lászlónak, hogy segített kiigazodni a tudományos publikációk útvesztőjében, s az időt, amit cikkeim és dolgozatom átoltvánasára rászánt. Köszönettel tartozom kutatásomban részt vevő hallgatóimnak és szerzőtársaimnak. Lelkesedésük és elszántságuk segített tüllépni a tévutakon és zsákutcákon, melyek sajnos az alap kutatás természetes velejárói.

Köszönöm Bokor Mihálynak, Kolumbán Sándornak és Lukovszki Tamásnak a tanácsaikat, melyek sokat segítettek bizonyításaim tökéletesítésében.

Köszönöm Agárdi Tamásnak, a Grafoló giai Intézet elnökének segítségét és a hasznos konzultációkat, melyek megalapozták kutatásom hátterét.

És végül, de nem utolsó sorban, köszönöm Dóra Melinda Tündének a türelmét, mellyel dolgozatom korrektúrázásában segített.
1. Introduction

The verification of handwritten signatures is one of the oldest biometric identification methods. As it gained a high legal acceptance, both the methods of forgers and verifiers became more elaborate. At the beginning of the 20th century, anyone who was in some way professionally connected to handwriting (teachers, notaries) could be treated as handwriting experts [1]. Today, however, being a forensic document examiner is an independent profession, which requires special training and a rich technological toolkit. There are even some guides [2] to modularize and organize the human workflow of the verification process. Despite these facts, the task of signature verification is still challenging, even for a human expert.

When confronted with professionally forged signatures, the average error rate of a forensic expert lies between 0.5% and 7% [3] while non-experts’ results may be much worse [4] (error rates between 10% and 26% were measured). These numbers show that even the opinions of forensic document examiners are prone to human errors and that there are still many open questions in the field. Questions such as: Why is it that the experts were unable to achieve better results? Are the previous numbers due to the limits of the verifiers, or the limits of the discriminative power of handwritten signatures? Now, however, rapid advancement of computer sciences has made it possible to analyze some of these questions more thoroughly.

The aim of computer-based signature verification is to automatically decide whether a given signature (questioned signature) belongs to a given person. The decision must only be based on some samples (original signatures) from the signer. Depending on the format of the samples, the field can be divided into two main categories, the on-line (or dynamic) and the off-line (or static) approach.

In on-line signature verification, the whole process of signing is captured using some kind of an acquisition device (a camera, a digitizing tablet, a stylus-operated PDA etc.), then analyzed and used to make a decision (Figure 1.1). The captured information usually includes pen position, pen pressure, pen azimuth and pen inclination as a function of time. This set of data gives automatized verification two important advantages. First, because data is available as a function of time, it is easier to identify corresponding parts, second, this kind of acquisition records information which is not directly available to the forger (like pen velocity, or pressure). Even when producing an almost perfect visual copy of a signature, these invisible factors will usually significantly differ from the
values measured during the original signing process. This is why state of the arte on-line signature verification systems can deliver error rates below 1% [5].

On the other hand, the aim of off-line signature verification is to decide, whether a signature originates from a given signer merely based on the image of the questioned signature and a few images of the original signatures of the signer. This means the input is limited to two-dimensional images while important pieces of information like velocity, inclination or pressure are mostly lost. Unlike on-line signature verification, which requires a special acquisition hardware and setup, off-line verification can be performed separately from the normal signing process and is thereby less intrusive and more user-friendly. Moreover, the off-line scenario can have a much wider range of practical applications as it can be seamlessly fit into many existing workflows. For example, the majority of financial institutions already digitize their contracts, mail or transfer orders, therefore adding automatic signature verification to their workflows would require “just” a software upgrade without directly affecting any of the clients and most of the employees.

While on-line signature verification has a definite advantage over human experts because of the captured non-visual information, off-line verification has a disadvantage, because it works with the same input, but does not have the extensive background knowledge of humans. This is the main reason why even the best off-line signature verifiers on the most studied databases cannot break the 9% error barrier [6] [7]. This limits their practical applicability significantly.

Several different approaches exist to overcome these limitations, but most of the signature verification systems have one thing in common. They behave mostly like black boxes (mainly due to the software’s AI-based approach) providing only little meaningful information about the reasons for a decision [8]. This makes their improvement a hard task.

In this work, we introduce an approach more similar to the federal document examiners (FDE) [2] who compare specific features of a signature individually to make their final decision. This approach also allows us to create a probabilistic model for our verification system. Using this probabilistic model one is not only able to create a classifier for our verification system, but also to make a good estimation about the accuracy of the system. For example, one is able to directly calculate how a change in the number of original samples, or the quality of forgeries will influence the final classification results without actually performing the tests.

1.1 Thesis contributions

In the next sections we propose a formal approach for off-line signature verification. By breaking down the process into smaller, loosely coupled tasks, and by ensuring a controlled information flow, it allows us to separately model and benchmark our components, and to identify and measure the different sources of errors. In contrast to usual monolith approaches [8] it not only allows us to better understand the weaknesses of a given verification approach but also provides means to predict the performance of a system based on parameters of the database.
The objective of my research was to establish a signature model as well as new evaluation methodologies, suitable for differentiating between original and forged signatures. Taking these aspects into account, my goals – addressed by this thesis – were the following:

- Investigate the common characteristics in existing off-line signature verification approaches and identify their strong and weak points
- Examine and – when necessary – improve existing feature extraction algorithms
- Identify the dominant features of signatures to be used to discriminate between original and forged ones.
- Establish and investigate classification techniques which may be used to differentiate between original and forged signatures
- Apply the methods to common signature databases and demonstrate the feasibility of practical application

1.2 Thesis structure

The rest of this thesis is organized as follows.

- Chapter 2 is devoted to illustrate the motivations of the results presented in this thesis. This chapter serves as a general introduction to the topic of off-line signature verification. It also outlines the most important approaches in the field. A modular and streamlined signature verification architecture is introduced and adapted to feature based verification. In the closing part of the chapter, we examine some of the major signature corpuses which were also used to test our algorithms.
- In Chapters 3 and 4, after reviewing the most relevant related work, we present the major contributions of this thesis. A statistical model is introduced, which is able to describe several important aspects of a signature database and predict their effect on the accuracy of the classification phase. Both the basic assumptions and the effects of the model are validated through a number of experiments.
- Chapter 5 summarizes the improvements we made to existing feature-extraction and -matching algorithms, which contributed to the results introduced in the previous chapters.
- To illustrate the practical relevance of this research, the application of the results is shown in Chapter 6. In the past years, a rich infrastructure has been built around our signature verification core. This chapter briefly introduces some of the most important ones.
- The last chapter (Chapter 7) is devoted to the summary and the outline of future work.
- In the Appendix, some detailed proofs are given, along with sample signatures from the major signature databases.
2. Motivation and background

This chapter introduces the basic concepts of off-line signature verification and gives an overview of the current state of the art.

2.1 Terms and concepts

As automatized signature verification is still in development, some terms in the field may have ambiguous interpretations. To avoid miscomprehension, throughout this thesis we are going to use the following interpretations of basic terms.

**Signature:** A person's name written as a form of identification in authorizing a document.

**Signer:** The creator of a signature.

**Original signature** (genuine signature): a signature originating from a given signer.

**Forged signature** (forgery): a signature which imitates the signature of a given signer but does not originate from him/her. Forgeries can be divided into several subclasses

- **random forgery:** a signature, with negligible similarities to the reference signatures. In common testing scenarios a random forgery is just a randomly chosen signature of another signer,
- **simple forgery:** a forgery, which was created with the knowledge of the name of the original signer, but without access to the reference signatures,
- **synthetic forgery:** an artificially generated signature, which was created by distorting some features of the reference signatures,
- **skilled forgery:** a forgery created after a thorough study of the reference signatures,

**Reference signature:** a signature of a signer known to the verification system. This kind of signature is used to train a verification system.

**Sample signature** (sample, questioned signature): a signature whose signer is not known to the verification system. This kind of signature is used to test a verification system.

**Original sample:** an original signature used as a sample.

**Forged sample:** a forged signature used as a sample.

**Accepted sample:** a sample, which was classified as an original signature by the verification system.
**Rejected sample**: a sample, which was classified as a forgery by the verification system.

**Inconclusive sample**: a sample upon which the system was unable to make a decision about the origin of the signature.

### 2.2 Performance metrics

As most biometric authentication systems, the performance of signature verifiers is usually measured in terms of error rates [9] [10]. Different error rates can be derived from statistical hypothesis testing, where the null hypothesis is:

$$H_0: \text{"The sample signature originates from the same signer as the reference signatures"}$$

and the possible decisions are a definitive “accept” or “reject”. We should note that in practical applications an “inconclusive” answer may be also allowed, but in our case this would severely limit the comparability of different verification systems.

**Definition 2.1. (Type I error):**

A type I error, also known as error of the first kind occurs when the null hypothesis ($H_0$) is true, but is rejected. In signature verification, the rate of type I error is often called False Rejection Rate (FRR) and is denoted by $\alpha$.

$$\alpha = \frac{\text{number of rejected original samples}}{\text{total number of original samples}}$$

**Definition 2.2. (Type II error):**

A type II error, also known as an error of the second kind, occurs when the null hypothesis ($H_0$) is false, but it is erroneously accepted as true. In signature verification, the rate of type II error is often called False Acceptance Rate (FAR) and is denoted by $\beta$.

$$\beta = \frac{\text{number of accepted forged samples}}{\text{total number of forged samples}}$$

**Definition 2.3. (Equal error rate):**

Each verification system can be tuned to a level at which both accept and reject error rates are equal. This rate is called equal error rate ($EER$).

$EER$ provides a quick way to compare the accuracy of verification systems. In general, a system with lower $EER$ is considered to be more accurate.

**Definition 2.4. (Average error rate):**

Average error rate ($AER$) is the average of $FAR$ and $FRR$ for a given experimental configuration.
In several cases, it is not feasible or possible to calculate or measure the exact EER (e.g. some papers only provide FAR and FRR measurements). For most systems considered in this thesis, the difference between the reported FAR and FRR values is small (mostly below 5%), therefore AER may be used as an admissible approximation of EER.

As mentioned before, we have deliberately ignored the possibility of an “inconclusive” decision in the previous statistics. Moreover, the notion of EER or AER would not be fit to describe the abilities of a real world application, where the focus is biased towards keeping either FAR or FRR low. These simplifications (namely making definite decision and considering both FAR and FRR with the same weight) are the de facto standards for comparing the performance of signature verification systems in the scientific literature.

The introduction of inconclusive decisions could significantly decrease the FAR and FRR values, but it would also render their interpretation ambiguous and would make their comparison more problematic.

On the other hand, although AER or EER may describe the performance of a verification system in general, they do not provide information about how they may perform, when they are tuned to make stricter or more tolerant decisions. To compensate for the latter, several papers also provide the receiver operating characteristic (ROC) curve of their systems, which is able to describe several FAR-FRR pairs, achievable by their systems. However, these ROC curves are hard to compare without more knowledge about the concrete verification problem, which has to be solved by the system.

### 2.3 The human factor

Signature verification is a behavioral biometric authentication method. This means that unlike fingerprints or iris patterns, which cannot be influenced by the subject, signatures are the results of conscious human behavior and can thereby be influenced by a wide range of factors [11].

On the positive side, this allows the seamless combination of biometric and knowledge-based authentication. While we currently limit our studies to written names, these algorithms could be easily extended to process any word, or combination of words written in the own handwriting of the person.

On the negative side, the conscious nature of the signing process can result in a wide range of variations in the signature of the same person. Fatigue, mood, glasses, pen, writing position and even the alignment of the form can affect the details of the signature. The basic assumption of signature verification is that there are some unconscious features, which stay mostly stable despite the previously mentioned factors. Moreover, some of these features (like the tremor of strokes) are extremely hard to forge by freehand methods. The main challenge is to find algorithms that can efficiently and universally locate, extract and compare such features.
2.4 Artificial forgeries

There is a wide variety of methods to forge signatures with artificial aids. Modern photocopiers or special plotters can create forgeries that are perfectly indistinguishable from the original signature to the naked eye. These methods always leave behind traces easily recognizable at a higher magnification rate, or by the analysis of color histograms. Our thesis will therefore be limited to freehand forgeries by humans.

2.5 State of the art

At the First International Signature Verification Competition [12], 16 different on-line signature verification systems were evaluated under similar conditions, among them those of the leading research groups in the field. The best system delivered an EER of 2.8% while the second best achieved 4.4%. Some studies suggest, that even better error rates can be achieved such as 0.35% in [13] or 0.4% in [14], however, these systems were evaluated on different datasets.

The off-line verification of signatures is even harder to tackle. While the on-line problem can mostly be reduced to matching and comparing one-dimensional functions (pen velocity, vertical, horizontal positions and all other measurements can be expressed as functions of time), the input for the off-line decision is only a single two-dimensional function, describing the intensity (or the color components) of a given pixel at a given vertical and horizontal position for an image. On-line capturing devices are also able to record additional information like pen pressure, azimuth, and altitude, which have been shown [15] to improve the final accuracy of a system.

Lacking a common signature corpus and a well-defined evaluation methodology, the results of different studies are difficult to compare; therefore, reported error rates should be only taken as approximations.

To highlight some common problems of comparison, we should consider the following papers. In [16] the authors achieved an FRR of 3% using a neural network classifier, but they only used 160 signatures from 10 signers. In addition, they only performed a recognition task (their system had to assign each of the 60 test signatures to one of the previously known signers). [17] reports an EER of 1% on random forgeries, however, this value drops to 19.8%, when using skilled forgeries. To allow for a mostly uniform comparison of different systems and the filtering of results irrelevant for our thesis, we shall formulate the following preconditions:

- A verification system should not take advantage of the fact that it has knowledge of all possible signers, as this is not the case in real world scenarios.

- We do not have any forgeries for a given signer during the training. Therefore the training phase should not take advantage of human-created forged signatures (synthetic forgeries are allowed)
- We assume that a system that is able to detect skilled forgeries will perform even better on simple or random forgeries. Therefore, we limit our analysis to skilled forgeries.

Based on the surveys, which together cover more than 500 papers from the last 30 years [5] [18] [19] [20] [21] and on [6] [7] covering some of the latest developments, it is safe to conclude that the best equal error rate currently achievable by off-line signature verifiers for skilled forgeries is approximately 9%. Some of the best results were achieved by [22] (FRR: 11.32% FAR: 6.48%), [6] (EER: 9.02% on GPDS-100 corpus and 8.8% on MCYT corpus\(^2\)) and [23] (EER: 10.65%).

To allow a better understanding of these results, they should be compared to results achieved by human experts. A 3 year long study of 28 FDEs, examining 11643 signatures (3387 originals and 8256 skilled forgeries) has shown, that human experts are able to confidently identify original signatures with a false rejection rate below 7% [3]. Unfortunately, this study also allowed “inconclusive” answers, therefore the evaluation of FAR was not possible. In [24] a study was performed on 432 genuine and 333 forged signatures from 51 writers, concluding that (non-expert) humans performance is FRR: 9% and FAR: 10.5%. In other studies experts achieved 0.5% FAR and 7% FRR, compared to the 6.5% FAR and 26% FRR of a layperson [25] [4].

The large difference between human performance and the performance of off-line signature verifiers suggests that, there is space for improvement. In the next section, we examine the structure of major existing verification systems and identify the possible means of improvement.

2.6 The verification process

The majority of signature verification methods can be divided into five main phases: acquisition, preprocessing, feature extraction, processing and classification (although these steps are not always separable). In an off-line scenario data acquisition is simply the scanning of a signature. This is followed by preprocessing, whereby the images of signatures are altered (cropped, stretched, resized, normalized, etc.) to create a suitable input for the next phase.

The next step is feature extraction, the process of identifying characteristics inherent to the particular person. The processing phase prepares the raw data extracted from the signature for classification. Using these results, the classification phase is able to make a decision whether to accept or reject the tested signature (Figure 2.1). This coarse separation of processing phases is already an extension to [4] which does not separate feature extraction from processing and classification. In our model, even further extensions will be necessary to allow a better control of the dataflow. In the following subsections, these 5 steps will be explained in detail and matched to the steps of several other signature verifiers.

\(^2\) Signature databases are discussed in detail in section 2.10.
2.6.1 Acquisition

In general, the acquisition step converts a number of paper sheets to a set of digital images, each of them containing one or more signatures. It is essential to note that scanning paper sheets with written signatures is not the only way to acquire digital images. As noted in [27], samples can also be generated from on-line databases or by altering existing signatures [28]. Of course, these latter methods cannot be used to validate the whole signature verification system; however they usually contain valuable additional information (like the correct order, direction and position of strokes), and this can be used to benchmark separate parts of the system. In most of the observed systems, the digitalized images of the signatures are assumed to be present, therefore the acquisition phase is usually not part of system diagrams.

2.6.2 Preprocessing

The preprocessing phase is a sequence of image transformations creating the best possible input for feature extraction algorithms. In on-line signature verifiers, the acquired data is usually already in an optimal form for further processing, therefore this phase is superfluous [29] [30]. In the off-line case it is usually necessary to eliminate the noise introduced during the acquisition phase [31]. Some preprocessing steps such as noise filtering, rotation normalization and position normalization induce only minimal information loss, while others, like binarization, morphological closing or size normalization can cause the loss of valuable information. Thus the second class of preprocessing steps is only applied where the feature extraction algorithm directly benefits from them. Although preprocessing is used in all of the examined off-line verifiers, only three of them [26] [32] [33] display it on their system diagram.

2.6.3 Feature extraction

“An image feature is a distinguishing primitive characteristic or attribute of an image. Some features are natural in the sense that such features are defined by the visual appearance of an image, while other, artificial features result from specific manipulations of an image. […] Image features are of major importance in the isolation of regions of common property within an image (image segmentation) and subsequent identification or labeling of such regions (image classification).” [34]. Therefore feature extraction is the location and characterization of features, and generally it should not be confused with later processing phases. Contrary to preprocessing, which is defined as a sequence of transformation steps altering the original images, feature extraction is a set of (usually) independent functions returning a characteristic feature set for their input image. Several systems (like those on Figure 2.2) take advantage of
multiple features to improve the quality of the input provided for distance calculations and classifiers.

Features can either be particular to the whole signature (global features) or to a part of the signature (topological or local features) [35].

![Figure 2.2: Using multiple features for the decision. Sources: [36] [37]](image)

### 2.6.4 Processing

The processing phase differs from the previous ones in that it can work with multiple images. First, a matching is defined between the features of the different images, then a distance (or similarity) measure is calculated based on the corresponding features and finally this measure may be normalized to make it a suitable input for a classifier. All of the three steps can be identified in Figure 2.1, while the other systems do not mention all of them. This can be explained by a different interpretation of phases, for example the feature extraction boxes usually also represent the processing phase (Figure 2.2) and score normalization is sometimes considered to be part of the classification phase.

### 2.6.5 Classification

In the classification phase, a single classifier is trained with reference signatures. Based on the training, the classifier can make decisions about the acceptance or rejection of a sample signature (Figure 2.3).

The complexity of classifiers can vary from checking a single threshold by using a hidden Markov model (HMM) to involving a neural network or Support Vector Machine (SVM) in the decision. These methods can also be combined in a composite system to allow the decision to be made with a deeper understanding of the context.

From the aspect of applicability it is essential to see whether the classification methodology only requires the presence of original signatures (a one-class problem) or both originals and forgeries (a two-class problem). The latter usually generates better results [38] [39], especially for small databases, but in “real world” scenarios, it can only be used if synthetic forgeries for each signer are provided.
There seems to be a major trend towards using general classifiers, like neural networks or SVM instead of concentrating on semantically interpretable distance measures, like the difference of local features (Figure 2.4). One major drawback of this is the increase in the number of two-class-problems, as these systems require both negative and positive samples for training.

2.7 Off-line signature verification systems

Table 2.1 summarizes some of the best off-line signature verifiers introduced in the last two decades. To allow a better understanding of the results, we also briefly describe the algorithmic approach as well as the signature databases used by these systems. A detailed description of the databases can be found in section 2.10.
Table 2.1 Comparison table of off-line signature verifiers

<table>
<thead>
<tr>
<th>Year</th>
<th>EER (rand)</th>
<th>Processing</th>
<th>Classification</th>
<th>Classes</th>
<th>Signers</th>
<th>Original signatures</th>
<th>Skilled forgery (simple forgery)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[41]</td>
<td>1995</td>
<td>(9.14)</td>
<td>Fuzzy ARTMAP</td>
<td>one</td>
<td>5</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>[42]</td>
<td>1996</td>
<td>14.3</td>
<td>Structuring elements</td>
<td>Linear classifier</td>
<td>one</td>
<td>55 550</td>
<td>300 (100)</td>
</tr>
<tr>
<td>[43]</td>
<td>1997</td>
<td>(0.84)</td>
<td>Shape factor, shape matrices</td>
<td>Distance function</td>
<td>one</td>
<td>20 800</td>
<td></td>
</tr>
<tr>
<td>[44]</td>
<td>1999</td>
<td>24.5</td>
<td>Extracted displacement fn.</td>
<td>Threshold</td>
<td>one</td>
<td>20 200 200</td>
<td></td>
</tr>
<tr>
<td>[45]</td>
<td>1999</td>
<td>(14)</td>
<td>HMM</td>
<td>HMM</td>
<td>one</td>
<td>60 1440</td>
<td></td>
</tr>
<tr>
<td>[46]</td>
<td>2000</td>
<td>(0.9)</td>
<td>Horizontal, vertical segmentation</td>
<td>HMM, majority vote</td>
<td>one</td>
<td>100 4000</td>
<td></td>
</tr>
<tr>
<td>[47]</td>
<td>2000</td>
<td>9**</td>
<td>Edges, directions, model</td>
<td>Thresholds</td>
<td>one</td>
<td>10 50 200 (100)</td>
<td></td>
</tr>
<tr>
<td>[48]</td>
<td>2002</td>
<td>5.9</td>
<td>Columns as vectors</td>
<td>HMM</td>
<td>one</td>
<td>100 4000 1200</td>
<td></td>
</tr>
<tr>
<td>[49]</td>
<td>2002</td>
<td>18.1</td>
<td>Projection, DPM</td>
<td>Threshold</td>
<td>one</td>
<td>55 CEDAR 1320 1320</td>
<td></td>
</tr>
<tr>
<td>[50]</td>
<td>2003</td>
<td>9.1</td>
<td>Modified Pattern Matching</td>
<td>Threshold</td>
<td>one</td>
<td>10 1000 1000</td>
<td></td>
</tr>
<tr>
<td>[39]</td>
<td>2004</td>
<td>9.3</td>
<td>Geometry, strokes, topology</td>
<td>SVM</td>
<td>two</td>
<td>55 CEDAR 1320 1320</td>
<td></td>
</tr>
<tr>
<td>[39]</td>
<td>2004</td>
<td>21</td>
<td>Geometry, strokes, topology</td>
<td>Distance methods</td>
<td>one</td>
<td>55 CEDAR 1320 1320</td>
<td></td>
</tr>
<tr>
<td>[51]</td>
<td>2004</td>
<td>9.6</td>
<td>Direct image</td>
<td>SVM (RBF)</td>
<td>one</td>
<td>40 160</td>
<td></td>
</tr>
<tr>
<td>[52]</td>
<td>2005</td>
<td>13.3*</td>
<td>Signature envelope shape</td>
<td>HMM</td>
<td>two</td>
<td>160 GPDS 3840 4800</td>
<td></td>
</tr>
<tr>
<td>[38]</td>
<td>2005</td>
<td>10</td>
<td>Density, moment, structure</td>
<td>multiple feed forward neural networks</td>
<td>two</td>
<td>30 300 150</td>
<td></td>
</tr>
<tr>
<td>[53]</td>
<td>2006</td>
<td>8.9</td>
<td>Modified Direction Feature</td>
<td>Neural Network (RBP, RBF)</td>
<td>two</td>
<td>39 GPDS 936 1170</td>
<td></td>
</tr>
<tr>
<td>[54]</td>
<td>2010</td>
<td>15.1</td>
<td>Gradient features</td>
<td>SVM (Gaussian kernel)</td>
<td>one</td>
<td>160 GPDS 3840 4800</td>
<td></td>
</tr>
<tr>
<td>[55]</td>
<td>2010</td>
<td>10.9</td>
<td>Gradient features</td>
<td>SVM (Gaussian kernel)</td>
<td>one</td>
<td>400 GPDS 9600 12000</td>
<td></td>
</tr>
</tbody>
</table>
### Motivation and Background

Table 2.1 shows that the major signature verifiers tend to work with global features like shape factors as well as different projections of the signature. Although this approach eliminates the need for complex matching algorithms, it also reduces the control over how individual parts of the signature are compared. Although using global features and advanced AI-based classification methods can yield error rates between 9% and 20%, it is hard to further improve these methods. We think that one of the main reasons behind this behavior is that the source of error is difficult to pinpoint. To break the 9% EER barrier, it is essential to identify and understand the different sources of error in the algorithms. Although such research has been conducted for on-line signatures \[15][56] the off-line scenario is essentially unanalyzed.

By using local (topological \[35\]) features, classification errors can mostly be traced directly back to specific errors in the processing chain that can be corrected, thereby improving the whole system. Several successful solutions in the two other major fields of biometrics – fingerprint and face recognition – are also based on matching some predefined features in the samples.

Based on the observations of the previous section, a generalized model for off-line signature verifiers is proposed \[57\].
The model in Figure 2.5 integrates the advantages of the previously introduced systems and their representations. Similarly to [58] and [33], it isolates the path of the training set (original signatures) from the path of test signatures while traversing the same modules of the system. The classifier is trained during the training phase only using the original signatures from the reference database, and classification decisions about test signatures will be made based on the training during the testing phase.

The model allows a straight data flow without feedback. This makes the loose coupling and individual testing of the components possible and allows the monitoring of error propagation (Appendix B).

As noted previously, individual phases can consist of multiple sequentially or parallel coupled subcomponents.

**2.9 The feature matching problem**

To allow the comparison of two signatures, the values extracted from one signature must be matched to the values of the other signature. This is a prerequisite for almost any classification approach. When working with global features (like width or height of a signature) this mapping can be defined easily. On the other hand, local features tend to be much more unstable (Figure 2.6). Some signatures of a signer may have 3 loops, while others may contain 4 loops. Without further preprocessing, this could result in input vectors of different lengths for the classifier. In Figure 2.5 the processing phase is represented by “feature matching” to emphasize the importance of this step. We are going to elaborate on this subject in section 5.1.3.

![Figure 2.6 Incorrect (left) and correct (right) matching of loops](image-url)
2.10 Test databases, preprocessing, experimental setup

Verification algorithms are developed and evaluated with, and tested against signature databases. A signature database (or signature corpus) is a set of digitalized signatures from a range of subjects. The database is commonly divided into a training set and a validation or test set, thereby eliminating the direct influence of the learning process on the verification process. A database can be characterized by a number of factors:

- **Number and type of signers**: these factors ensure the diversity of the database. To allow for a representative testing of algorithms, the number of signers should be as high as possible and not limited to a single social group (like students). Only a handful of public databases contain more than 100 signers. In [45] and [38], some of the experiments were repeated with an increasing number of signers (10-20-25-30 signers) and it always resulted in a worse EER value.

- **Number of original signatures**: this number should be high enough to ensure a set for training and a disjoint set for testing the signature verifier.

- **Quality of forgeries**: forgeries may be simple or skilled. Skilled ones are preferred, as their distinction from the original is a harder task.

- **Origin of forgeries**: Skilled forgeries may originate from one or from more forgers. The latter is preferred, as it increases the diversity of the test set.

- **Number of forgeries**: the number of forgeries should be in the same magnitude as the number of original samples in the test set.

- **Resolution**: an increased resolution allows better processing, while it also increases the required computational resource and storage space. Usual resolutions are 300dpi and 600dpi.

- **Color depth**: most of the public databases contain binary (black and white) images. Greyscale or color images increase storage requirements, but contain important information about the dynamics of the signing process.

In the following sections we introduce our test databases and some of the major public databases used or referenced in this thesis.

### 2.10.1 BME-AUT1

The BME-AUT1 [59] database is a collection of 1400 signatures. It contains 20 originals and 8 skilled forgeries for each of 50 signers. The 32-bit color images were scanned at 600dpi. The forgeries are of unusually good quality as they were all produced by handwriting experts. We used these samples during the first years of our research. The database is freely available for research purposes at https://www.aut.bme.hu/Signature.

### 2.10.2 BME-AUT2

The BME-AUT2 [59] database contains 50 original signatures from each 50 signers. This database contains no forgeries. The high quality (600dpi, 32-bit
color) images were used in the statistical analysis of our signature model. The database is freely available for research purposes at https://www.aut.bme.hu/Signature.

2.10.3 BME-AUT9
The BME-AUT9 is a smaller database mainly used in the fine-tuning of our feature extraction algorithms. It contains 16 original and 16 forged signatures for each of the 9 signers. The images are available at 600 dpi resolution with 32-bit color depth. As the signers are teachers from the university, this database is not available for public use.

2.10.4 SVC2004
The SVC2004 is a public signature corpus, created for the First Signature Verification Competition [12]. This is an on-line signature database; therefore, it contains stroke information without the images themselves. The stroke information was used to synthesize signatures similar to the original ones. Stroke points were connected with straight lines, fading out on the line borders. Bicubic interpolation and anti-aliasing were used to make the final image smoother. Eight hundred Latin-type signatures from 20 signers (20 originals and 20 forgeries from each) were used to test our feature extraction and classification algorithms. Appendix C contains a range of image samples from this database.

2.10.5 GPDS
The database created by Grupo de Procesado Digital de Señas at the University of Las Palmas in Gran Canaria, Spain is the largest public corpus for off-line signature verification [60]. Signatures were collected from 960 different signers. 24 original and 30 forged signatures ensure enough samples for both training and testing different verification systems. As the database was created incrementally, several publications may refer to GPDS100, or GPDS300 meaning thereby the first 100 or 300 signers of the database. Although the range of signers is very wide, the 300dpi binary images are of very low quality (Figure 2.7).

Also, the majority of the signatures contain wide loops covering a larger part of the signature. Such features are (according to our experience) not representative for the global population. Such signatures favor global feature-based approaches, while local feature-based approaches could get confused by the large variety of line crossings.

Figure 2.7 Samples from the GPDS database
2.10.6 MCYT
The MCYT is a multimodal biometric database containing fingerprint and signature data from 330 contributors collected on 4 different Spanish sites [61]. The MCYT-75 off-line signature corpus is a publicly available subset of the database containing images for 15 original and 15 forged signatures of 75 signers. The 2250 images were scanned at a resolution of 600 dpi.

2.10.7 CEDAR
The database [39] of the Center of Excellence of Document Analysis and Recognition at the University at Buffalo, the State University of New York is a collection of 2640 signatures containing 24 original and 24 forged samples from 55 signers. The 600dpi greyscale images are publicly available.
3. The statistical model of signature verification

We have already discussed in section 2.6.5 that there is an overwhelming trend of using AI-based classifiers. The reason behind this is simple: we do not know exactly what we are looking for. The basic assumption behind most of these approaches is that the feature values of original signatures will cluster around specific means and these clusters will be separable from the feature values of forgeries. Although these assumptions are mainly confirmed by the experimental results, a deeper investigation of the reasons behind it is rarely conducted.

In this section, we review the related work and establish the basic terminology, next we present and refine a novel classification approach, purely based on the statistical characterization of features. All theoretical results are presented in conjunction with their experimental validations.

3.1 Related work

As an example for statistical methods, in [62] the Kolmogorov-Smirnov Test is used for obtaining a probability of similarity between the distribution of the reference signatures and the sample signatures in distance space. The achieved error rate was 16.4% on the CEDAR database, but no further efforts were done to refine the model or to examine the distributions of feature values.

A more thorough study of the statistical approach was presented in [63]. The basic assumption of the paper is that when measuring any physiological characteristic of a person\(^3\), the values will be distributed according to the normal distribution \(\mathcal{N}(\mu_i, \sigma_i^2)\) and that the entire population is also similarly distributed but with parameters \(\mu_p\) and \(\sigma_p\). We can use this to calculate the statistical probability of a given sample belonging to a given person. A key element influencing the accuracy of the system is the proportion of \(\sigma_i\) and \(\sigma_p\), which they described with \(\sigma = \sigma_i / \sqrt{\sigma_i^2 + \sigma_p^2}\).

Our approach is very similar, but we made three major improvements. First, we focus on verification instead of identification. Therefore, while we do not take advantage of the knowledge of the population, we assume knowledge of the distribution of forgeries \(\mathcal{N}(\mu'_i, \sigma_i'^2)\). Second, we will use the quotient \(q = \sigma_i / \sigma'_i\) to describe the relation of standard deviations, which is in our opinion a more intuitive descriptor for the quality of forgeries. Third, and most importantly, we will show that a low sample size (the paper mentions sample sizes between 3 and 5) significantly influences the accuracy of classification and should therefore be incorporated into calculations. In addition, we will perform several tests of the assumption of normally distributed variables.

\(^3\) The paper was not directly focused on signature verification, but considered all kinds of biometric authentication and identification.
It is also widely accepted that the accuracy of a biometric system can be improved by increasing the number and diversity of features [64]. Several approaches (like [23]) have significantly improved this way, but the effect of the number of features on the final results is always established empirically. We are going to show that as long as the features remain mostly independent, their effect on the final results can be modeled and estimated without actually performing the verification.

3.2 The terminology of features

After acquisition and pre-processing, several features are located and their properties are extracted. Feature properties are quantitative descriptors of different aspects of the signature (e.g., height of the first loop, pitch of the signature, etc.). We refer to a class of similar features as “feature types”. For example, “loop” is a feature type, as it represents a generic characteristic element of signatures. Instances of a feature type will be called “features” (e.g., in Figure 2.6, there are four loops in a signature). A feature is described by several quantities (a loop can have a height, a width, an area, etc.). These quantities will be called feature properties.

A special group of feature types is characteristic for the whole signature (width, pitch, etc.). These feature types within the group are called global feature types. All of the other feature types, which may have multiple instances within a single signature (e.g., loops or line crossings), are called local feature types (Figure 3.1). Common signature verification approaches may take advantage of both global and local types [36].

In the subsequent sections the following definitions are used:

**Feature**: a feature is a part of the signature described by a set of properties. We distinguish two types of features:

- **global feature**: a feature which covers the whole signature
- **local feature**: a feature which only covers a smaller part of the signature

**Feature property**: a number describing a given aspect of a feature

**Feature type**: describes a group of features that have the same properties
Although there is a huge repertoire of feature types in the literature, our research focuses only on the ones that are easily interpretable by humans. This process most closely matches the FDEs approach and fosters a better intuitive understanding of the final results. In our current experiments, two feature types are included: baselines and loops. Both types were chosen and defined using the knowledge gained in several consultations with handwriting experts [8].

Our current approach is based on the statistical characterization of feature properties. First, we introduce a method for a single feature property, analyze and confirm our assumptions, and then we generalize the method to multiple feature properties.

### 3.3 The distribution of feature properties in original signatures

**Assumption 3.1**

A feature property can be approximated as a normally distributed random variable $X$, with mean $m$ and variance $s^2$

$$X \sim \mathcal{N}(m, s)$$  \hspace{1cm} (3.1)
Remark: It is natural to assume that the signer is aiming to reproduce his own signature as accurately as possible; therefore, a given feature property will most likely vary around a given mean. Additionally, the exact result is influenced by a large number of different independent factors. These conditions in natural sciences are usually enough to warrant the use of a normal distribution to (at least approximately) characterize a variable. This assumption was also used in [63]. In the next sections, we demonstrate with several experiments that this approximation is sufficient for the purposes of signature verification.

3.4 Experiments

The main purpose of this experiment is to support the assumption of normally distributed feature properties. The experiments were performed on 20 original signatures from each of the 20 signers from the SVC2004 database. During the feature extraction and matching steps, 35 baselines and 37 loops were identified from the original signatures of signers. Baselines consisted of 6 properties and loops consisted of 14 properties.

Two different normality tests were selected to check Assumption 3.1. The Lilliefors adaptation of the Kolmogorov-Smirnov test [65] and the Shapiro-Wilk test [66] are both known to perform well on low sample sizes; therefore, all of the data series were tested with both of these tests. Figure 3.2–Figure 3.5 provide a simplified view of the results. The results are grouped by feature type (e.g., Figure 3.2 and Figure 3.3 show the results for the baselines). Each column represents a baseline in a signature, and each row represents a feature property (e.g., X position) of the baseline. If a given cell is dark, the given property of the given feature did not pass the normality test (the null hypothesis was rejected with a 0.05 significance level).

![Figure 3.2. The Shapiro-Wilk test for baselines](image)

![Figure 3.3. The Lilliefors test for baselines](image)
As a side effect of our trimming algorithm, the horizontal position (x1) of the first baseline was almost always chosen to be a fixed value. Therefore, it almost always failed the tests (Figure 3.2 and Figure 3.3). Some invalid matches could also be identified by the increased number of errors in a given column (columns 2, 10, and 26 on Figure 3.2 and Figure 3.3 and columns 15 and 16 in Figure 3.4 and Figure 3.5). These matching errors were also confirmed by a visual analysis of the signatures.

Altogether, if the first parameter of the baselines was omitted, the normality tests would have only failed in 20% of the cases for the Shapiro-Wilk test and in 22% of the cases in the Lilliefors test. Yet a significant portion of errors is explained by errors introduced previously in the processing phases. Visual inspection showed that 90% of the errors were introduced because of an erroneous matching of features or errors in the feature extraction algorithms. Only 10% of all of the failed normality tests can directly be explained by an irregular distribution of feature attributes. Measurement errors, sampling errors, low sample sizes or unusually irregular features may account for the latter. Although these results should not be considered a direct proof of normality, we consider these results encouraging enough to use Assumption 3.1 to approximate the characteristics of feature properties.
3.5 The distribution of feature properties in forged signatures

Assumption 3.2

A forged feature property can be approximated as a normally distributed random variable $Y$.

$$Y \sim \mathcal{N}(m_f, s_f)$$  \hspace{1cm} (3.2)

In addition, because the forger is aiming to forge the original signature, the means of these distributions overlap.

$$M(X) = M(Y) \rightarrow m_f = m$$ \hspace{1cm} (3.3)

Therefore, the only difference between original and forged feature values is in their deviations. We defined the quality of a forgery ($q$) as their quotient.

$$q = \frac{s_f}{s}$$  \hspace{1cm} (3.4)

For the rest of this thesis, we are going to use the lower index $o$ to annotate functions or variables belonging to original signatures and the lower index $f$ for forged signatures.

Remark: The first statement of Assumption 3.2 is a corollary of Assumption 3.1. As forgers aim to reproduce a given feature, the resulting properties will vary around a given mean. Although a single forger may make the same systematic error, when considering several possible forgers, these deviations should cancel themselves out resulting in the same expected value as in the feature properties of the original signatures (Eq. 3.3).

3.6 Experiments

To support Assumption 3.2 several measurements were performed on the GPDS300 database (using all 24 original and 30 forged signatures from the 300 signers). The average and the deviation of feature properties both for original ($\bar{m}_o, \hat{s}_o$) and forged ($\bar{m}_f, \hat{s}_f$) signatures for each of the signers have been calculated and their average quotient is summarized in Table 3.1, Table 3.2 and Table 3.3. These tables show, that while the averages of original and forged values are relatively close (the difference is mostly below 10% and in several cases below 1%), the quotients of deviations show a significant difference. It can be also seen, that the average value of $q$ (defined as: $\hat{s}_f / \hat{s}_o$) was 1.57 which – based on our experience – can be characterized as a typical forgery quality. It should also be noted, that the lowest value of $q$ in these tables was 1.1386, therefore in future calculations it is reasonable to assume, that $q > 1$, which intuitively means, that the forger cannot reproduce every aspect of the signature with the same accuracy as the original signer.
Table 3.1 The relation of original and forged baseline properties

<table>
<thead>
<tr>
<th>Feature property</th>
<th>$\hat{m}_f / \hat{m}_o$</th>
<th>$\hat{s}_f / \hat{s}_o$</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>1.0671</td>
<td>3.0789</td>
</tr>
<tr>
<td>Y</td>
<td>1.0532</td>
<td>1.4792</td>
</tr>
<tr>
<td>X2</td>
<td>0.9963</td>
<td>1.6114</td>
</tr>
<tr>
<td>Y2</td>
<td>1.1024</td>
<td>1.5655</td>
</tr>
<tr>
<td>Angle</td>
<td>1.0072</td>
<td>1.6205</td>
</tr>
<tr>
<td>Length</td>
<td>0.9911</td>
<td>1.5967</td>
</tr>
</tbody>
</table>

Table 3.2 The relation of original and forged loop properties

<table>
<thead>
<tr>
<th>Feature property</th>
<th>$\hat{m}_f / \hat{m}_o$</th>
<th>$\hat{s}_f / \hat{s}_o$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perimeter</td>
<td>1.0979</td>
<td>1.5204</td>
</tr>
<tr>
<td>MaximumDiameter</td>
<td>1.0345</td>
<td>1.3857</td>
</tr>
<tr>
<td>MaximumDiameterAngle</td>
<td>1.3095</td>
<td>1.6218</td>
</tr>
<tr>
<td>InscribedDiameter</td>
<td>1.1585</td>
<td>1.4510</td>
</tr>
<tr>
<td>BoundingCircleRadius</td>
<td>1.0337</td>
<td>1.3800</td>
</tr>
<tr>
<td>Area</td>
<td>1.5352</td>
<td>2.3323</td>
</tr>
<tr>
<td>MomentAxisAngle</td>
<td>1.0592</td>
<td>2.0204</td>
</tr>
<tr>
<td>Formfactor</td>
<td>1.1176</td>
<td>1.3034</td>
</tr>
<tr>
<td>Roundness</td>
<td>1.1716</td>
<td>1.3125</td>
</tr>
<tr>
<td>Compactness</td>
<td>1.0760</td>
<td>1.1956</td>
</tr>
<tr>
<td>Extent</td>
<td>1.0561</td>
<td>1.1531</td>
</tr>
<tr>
<td>Solidity</td>
<td>0.9994</td>
<td>1.6123</td>
</tr>
<tr>
<td>Convexity</td>
<td>1.0009</td>
<td>1.5233</td>
</tr>
<tr>
<td>AspectRatio</td>
<td>0.9157</td>
<td>1.1386</td>
</tr>
<tr>
<td>ModificationRatio</td>
<td>1.1594</td>
<td>1.2317</td>
</tr>
<tr>
<td>X</td>
<td>1.0039</td>
<td>1.8656</td>
</tr>
<tr>
<td>Y</td>
<td>1.0353</td>
<td>1.7148</td>
</tr>
</tbody>
</table>

Table 3.3 The relation of original and forged global properties

<table>
<thead>
<tr>
<th>Feature property</th>
<th>$\hat{m}_f / \hat{m}_o$</th>
<th>$\hat{s}_f / \hat{s}_o$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Width</td>
<td>0.9954</td>
<td>1.5484</td>
</tr>
<tr>
<td>Height</td>
<td>1.0432</td>
<td>1.4929</td>
</tr>
<tr>
<td>AspectRatio</td>
<td>0.9754</td>
<td>1.3410</td>
</tr>
<tr>
<td>TrimmedWidth</td>
<td>1.0046</td>
<td>1.5886</td>
</tr>
<tr>
<td>TrimmedHeight</td>
<td>1.0515</td>
<td>1.5007</td>
</tr>
<tr>
<td>TrimmedAspectRatio</td>
<td>0.9797</td>
<td>1.3443</td>
</tr>
</tbody>
</table>
3.7 Distinguishing original and forged feature properties

Let \( w \) be the threshold used to distinguish between original and forged feature properties using the following rule: each feature value that falls in the interval \([m - ws, m + ws]\) is accepted as an original, and each feature value which does not fall into the interval is taken as a forgery. This definition allows the unambiguous classification of a feature value as forged or original and introduces type I and type II errors (Figure 3.6).

![False Rejection Rate (\( \alpha \))](image1)

![False Acceptance Rate (\( \beta \))](image2)

Figure 3.6 The illustration of error rates

Error rates can be expressed based on the previous equations and \( w \).

\[
FRR = \alpha = P(m - ws > X) + P(m + ws < X) = 2 - 2\Phi(w) \quad (3.5)
\]

\[
FAR = \beta = P(m - ws < Y < m + ws) = 2\Phi \left( \frac{w}{q} \right) - 1 \quad (3.6)
\]

Provided that the a priori probabilities of receiving an original sample and receiving a forged sample are equal, the average error of such a system would be:

\[
AER = \frac{\alpha + \beta}{2} = \frac{1}{2} + \Phi \left( \frac{w}{q} \right) - \Phi(w) \quad (3.7)
\]

**Proposition 3.1**

\[
AER < \frac{1}{2} \text{ when } q > 1 \quad (3.8)
\]

**Proof:**

Because \( \Phi \) is monotonically increasing, and \( \frac{w}{q} < w \), when \( q > 1 \)

\[
\Phi \left( \frac{w}{q} \right) < \Phi(w) \Rightarrow \Phi \left( \frac{w}{q} \right) - \Phi(w) < 0 \quad (3.9)
\]

\[
AER = \frac{1}{2} + \Phi \left( \frac{w}{q} \right) - \Phi(w) < \frac{1}{2} \quad (3.10)
\]
We have shown that as long as the “quality” of forgeries is worse than that of original signatures, any threshold $w$ will result in a better classification result than a simple coin toss. In Proposition 3.2 we are going to show that this error rate can be minimized by the optimal choice of $w$.

**Proposition 3.2**

The sum of the probabilities of false rejection and false acceptance is minimized when

$$w = \pm \sqrt{2\ln q} \sqrt{\frac{q^2}{q^2 - 1}}$$

(3.11)

**Proof:** The error is minimized if

$$\left(\frac{1}{2} + \Phi\left(\frac{w}{q}\right) - \Phi(w)\right)' = 0$$

(3.12)

This is the case if

$$f_o(x) = f_f(x)$$

(3.13)

$$\frac{1}{\sqrt{2\pi s_o^2}} e^{-\frac{1}{2} \left(\frac{m-x}{s_o}\right)^2} = \frac{1}{\sqrt{2\pi s_f^2}} e^{-\frac{1}{2} \left(\frac{m-x}{s_f}\right)^2}$$

(3.14)

Using: $x = m + ws$ we get:

$$\frac{1}{s\sqrt{2\pi}} e^{-\frac{1}{2} \left(\frac{m-(m+ws)}{s}\right)^2} = \frac{1}{qs\sqrt{2\pi}} e^{-\frac{1}{2} \left(\frac{m-(m+ws)}{qs}\right)^2}$$

(3.15)

From which:

$$w = \pm \sqrt{2\ln q} \sqrt{\frac{q^2}{q^2 - 1}}$$

(3.16)

The detailed proof can be found in Appendix A.1.

In spite of the slightly different notation, these results can also be derived from the one obtained in [63] for the “single feature” scenario.

The above equations reflect only a theoretical scenario where both the mean and variance are known. In practice, this is not the case, as only the sample mean and the sample variance are available. Because of the usually small sample size (typically $n \leq 10$) this has a direct effect on the optimal choice of $w$. Therefore in Lemma 3.1 the distribution calculations for a feature-property are refined and based on the results in Proposition 3.3 the new value for $w$ is calculated by taking the sample size into consideration.
The Statistical Model of Signature Verification

Lemma 3.1

Based on the sample mean ($\bar{m}_n$) and sample variance ($\hat{s}_n^2$), and sample size ($n$), the probability distribution of a feature property can be calculated as follows:

$$t(x) = \frac{1}{\hat{s}_n \sqrt{\frac{1}{n} + \frac{1}{n}}} t_{n-1} \left( \frac{x - \bar{m}_n}{\hat{s}_n \sqrt{\frac{1}{n} + \frac{1}{n}}} \right)$$ \hspace{1cm} (3.17)

Where $t_{n-1}$ stands for the probability density function of the t-distribution with $n - 1$ degrees of freedom and $n > 1$.

Proof: Lemma 3.1 is proven in Appendix A.2

Proposition 3.3

Approximating the distributions with Student’s t-distributions, the error is minimized if

$$w = \pm \sqrt{\frac{q^2 - q^2 + 2}{2q^2 - q^2}} \left( \frac{n^2 - 1}{n} \right)$$ \hspace{1cm} (3.18)

Proof: Let $t_o(x)$ and $t_f(x)$ denote the probability distribution function (based on Equation 3.18) and $T_o(x)$ and $T_f(x)$ denote the cumulative distribution function of the original and forged samples, respectively.

In this case:

$$\alpha = P(m - ws > X) + P(m + ws < X) = 2T_o(\bar{m}_n - w\hat{s}_n)$$ \hspace{1cm} (3.19)

$$\beta = P(m - ws < X < m + w\sigma_o) = 1 - 2T_f(\bar{m}_n - w\frac{\hat{s}_n}{q})$$ \hspace{1cm} (3.20)

$$AER = \frac{\alpha + \beta}{2} = \frac{1}{2} + T_o(\bar{m}_n - w\hat{s}_n) - T_f(\bar{m}_n - w\frac{\hat{s}_n}{q})$$ \hspace{1cm} (3.21)

The error is minimized if

$$\left( \frac{1}{2} + T_o(\bar{m}_n - w\hat{s}_n) - T_f(\bar{m}_n - w\frac{\hat{s}_n}{q}) \right)' = 0$$ \hspace{1cm} (3.22)

Expanding the equation we get:

$$\frac{1}{\hat{s}_n \sqrt{\frac{1}{n} + \frac{1}{n}}} t_{n-1} \left( \frac{w\hat{s}_n}{\hat{s}_n \sqrt{\frac{1}{n} + \frac{1}{n}}} \right) = \frac{1}{q\hat{s}_n \sqrt{\frac{1}{n} + \frac{1}{n}}} t_{n-1} \left( \frac{w\hat{s}_n}{q\hat{s}_n \sqrt{\frac{1}{n} + \frac{1}{n}}} \right)$$ \hspace{1cm} (3.23)
From which:

\[ w = \pm \sqrt{\left(\frac{q^2 - q^2 + \frac{\pi}{2}}{q n - q^2}\right) \left(\frac{n^2 - 1}{n}\right)} \]  

(3.24)

The detailed proof can be found in Appendix A.3.

Equation 3.24 shows that the value of \( w \) is not independent of \( n \). This is a refinement of the estimation from [63], where the number of samples was ignored. The effect of sample size (\( n \)) on the optimal threshold choice (\( w \)) is illustrated in Figure 3.7. As sample size increases, the decision threshold converges to the threshold defined in [63]; however, at low sample sizes there is a notable difference in the results. This difference gets even more significant as the quality of forgeries decreases (as \( q \) increases).

![Figure 3.7 Optimal choices of decision threshold (w) based on the quality of forgeries (q) and the number of samples (n)](image)

Based on Eq. 3.24, Eq. 3.19 and 3.20 both \( \alpha \) and \( \beta \) can be calculated. See Appendix A.4 for details.

### 3.8 Building a classifier

Equation 3.21 describes the accuracy of a decision that is based on a single property of a feature. However, in real world scenarios, we are able to utilize several different feature properties. In this section, we first analyze the accuracy of a classifier built on the above principles using a single feature property, and then extend the model for multiple feature properties.

#### 3.8.1 Single feature property

Let the sample space \( \Theta = \{\theta_o, \theta_f\} \) denote the genuineness of the questioned signature, where \( \theta_f \) denotes the event when the signature is forged and \( \theta_o \) denotes the event when a signature is original. We have no a priori knowledge about the signature; therefore, the a priori probabilities of the events are taken as equal:
\[ P(\theta_o) = P(\theta_f) = 0.5 \] (3.25)

Let \( X \) be an observation of a given feature property, where \( X = 1 \) means that the observed feature property fell in the interval \([m - ws, m + ws]\) and was therefore accepted. Similarly, \( X = 0 \) means rejection. The probabilities of these observations in the case of a genuine signature are:

\[ P(X = 1|\theta_o) = 1 - \alpha \] (3.26)
\[ P(X = 0|\theta_o) = \alpha \] (3.27)

while in the case of a forgery:

\[ P(X = 1|\theta_f) = \beta \] (3.28)
\[ P(X = 0|\theta_f) = 1 - \beta \] (3.29)

In the case of a single feature property and a single observation where \( X = 0 \) (rejection), the probability that the signature is really a forgery can be calculated as follows:

\[ P(\theta_f|X = 0) = \frac{P(\theta_f)P(X = 0|\theta_f)}{P(\theta_f)P(X = 0|\theta_f) + P(\theta_o)P(X = 0|\theta_o)} = \frac{1-\beta}{1-\beta+\alpha} \] (3.30)

The error probability can be calculated as:

\[ P(\theta_o|X = 0) = \frac{\alpha}{1-\beta+\alpha} \] (3.31)

### 3.8.2 Multiple feature properties

The accuracy of the final decision can be further improved by performing several tests where each test is performed on a different feature property. If \( I \subseteq \{1 \ldots k\} \) tests identify the sample as a forgery, and \( \{1 \ldots k\} \setminus I \) tests identify the sample as an original (where \( I \in 2^k \)), and assuming the independence of features, the probability of the signature being a forgery can be calculated as:

\[ P(\theta_f|I) = \frac{\prod_{i\in I}(1-\beta_i)\prod_{j\notin I}\beta_j}{\prod_{i\in I}(1-\beta_i)\prod_{j\notin I}\beta_j + \prod_{i\in I}\alpha_i\prod_{j\notin I}(1-\alpha_j)} \] (3.32)

**Special case:** If for all tests \( \alpha_i \)-s are equal and \( \beta_i \)-s are equal, i.e., \( \forall \alpha_i = \alpha \) and \( \forall \beta_j = \beta \) and \( l = |I| \), we obtain the following equations:

\[ P(\theta_f|I) = \frac{(1-\beta)^l\beta^{k-l}}{(1-\beta)^l\beta^{k-l+\alpha}(1-\alpha)^{k-l}} \] (3.33)

Similarly, the probability that the signature is original is:

\[ P(\theta_o|I) = \frac{\alpha^l(1-\alpha)^{k-l}}{(1-\beta)^l\beta^{k-l+\alpha}(1-\alpha)^{k-l}} \] (3.34)

For simplicity’s sake, we take the previous assumptions as given in the next sections and use these simplified formulas for our classifications. Although
these two presumptions may seem restrictive, we are going to provide experimental proof, for both of them being justified. In Section 3.8.3 we are going to show, that it is possible to select feature properties which may be regarded as (mostly) independent, while in Section 4.3 we are going to show, that the errors introduced by the variability of \( q \) may be kept below 2% in most of the cases.

We should also note that the distribution of multiple features and thereby the decision mechanism could be probably more accurately modeled by a multivariate distribution. Although we intend to do so in our future works, our current simplifications are required in order keep the complexity of the proofs in Section 4 manageable. Moreover in section 3.8.3 we will show, that even with these simplifications our model is a good predictor of actual verification performance.

The previous two equations allow us to make a decision that minimizes the average error. If \( P(\theta_f|I) > P(\theta_o|I) \), the signature is classified as forged, and in the opposite case, it is classified as original. Note that \( \alpha \) and \( \beta \) only depend on \( n \) and \( q \). Figure 3.8 then shows how the value of \( l \) affects the certainty of the decision for a typical classification scenario (\( n = 10, q = 2.0, k = 20 \)).

**Figure 3.8 Probabilities of improper classification based on the number of rejected feature properties \((l)\)**

*Proposition 3.4:* If each of the feature properties is either rejected or accepted, and \( \alpha \) and \( \beta \) are known, the number of rejected features required to reject a signature while minimizing the average error rate is:

\[
l = k \frac{\ln(1-\alpha) - \ln(\beta)}{\ln(1-\alpha) - \ln(\beta) + \ln(1-\beta) - \ln(\alpha)}
\]  

(3.35)

*Proof:* We can find the point where

\[
P(\theta_f|I) = P(\theta_o|I)
\]  

(3.36)
The statistical model of signature verification

41

\[
\frac{(1-\beta)^l \beta^{k-l}}{(1-\beta)^l \beta^{k-l} + \alpha^l (1-\alpha)^{k-l}} = \frac{\alpha^l (1-\alpha)^{k-l}}{(1-\beta)^l \beta^{k-l} + \alpha^l (1-\alpha)^{k-l}}
\]

(3.37)

\[l = k \log \frac{(1-\alpha)(1-\beta)}{\alpha \beta} \frac{1-\alpha}{\beta} = k \frac{\ln (1-\alpha) - \ln (\beta)}{\ln (1-\alpha) - \ln (\beta) + \ln (1-\beta) - \ln (\alpha)}
\]

(3.38)

The detailed proof can be found in Appendix A.5

In this context, \(\lfloor l \rfloor\) describes the maximum number of rejected features that can be allowed in an original signature. It should be noted that the certainty of the decision increases as the value of \(l\) approaches \(k\) or 0. Figure 3.8 also shows that there is a greater uncertainty between 4 and 10 forged features. In a practical application this could be the interval of "inconclusive" decisions.

Using these results we can answer questions such as how many feature parameters must be classified as forged to make a decision about the origin of the signature with 99% certainty \((n = 10, q = 2.1, f = 20)\).

Proposition 3.5: The lowest achievable average error rate of a signature verification system can be predicted based on the number of reference signatures used to train the system \((n)\), the number of independent feature properties \((k)\) and the quality of the forged features \((q)\) as follows:

\[AER(q, n, k) = \frac{\sum_{i=0}^{\lfloor l \rfloor} \binom{k}{i} \beta^{k-i} (1-\beta)^i + \sum_{i=\lfloor l \rfloor + 1}^{k} \binom{k}{i} \alpha^i (1-\alpha)^{k-i}}{2}
\]

(3.39)

Proof: The total rates of false acceptance and rejection can be calculated from equations 3.33 and 3.34 by summing the error probabilities for all possible values of \(l\):

\[FRR(q, n, k) = \sum_{i=\lfloor l \rfloor}^{k} \binom{k}{j} \alpha^j (1-\alpha)^{k-j}
\]

(3.40)

\[FAR(q, n, k) = \sum_{i=0}^{\lfloor l \rfloor} \binom{k}{i} \beta^{k-i} (1-\beta)^i
\]

(3.41)

\[AER(q, n, k) = \frac{FRR(q, n, k) + FAR(q, n, k)}{2}
\]

(3.42)

Figure 3.9 shows the above three error measures as a function of \(k\), for \(n = 5, q = 2.0\).
Figure 3.9 Predicted error rates based on the number of examined independent feature properties (k).

Figure 3.9 also shows that (considering a fixed amount of original samples) at least 10 independent feature properties are required to achieve an error rate below 20%, 20 feature properties for 10% and further improvements require exponentially more properties, which may be why most verification systems fail to achieve error rates below 10%.

3.8.3 Experimental results

A classification module was created to test the precision of the original forgery decision, made only on the basis of our test database and using the previously introduced algorithms. Note that although several errors have been noted in the previous section, we did not alter our algorithms for this experiment.

The main purpose of the experiment is to show that our mathematical estimations can successfully predict the precision of a signature verifier. Therefore, several measurements were conducted with different original sample counts (n) and feature counts (k). During our experiments, the value for q was set to 1.6 for the SVC database and to 1.4 for the GPDS corpus, as the lower resolution and the higher complexity of signatures made the loop matching a harder task in the latter case.

Although the independence of features is a precondition of our theory, 100% independence can most likely never be ensured. To best match these preconditions, we have chosen (prior to the experiment) two global features, two attributes of loops and two attributes of baselines to approximate these conditions:

- aspect ratio (width/height) of the signature,
- moment axis of the signature,
- angle of the first baseline,
- length of the first baseline,
- angle of the second baseline,
- length of the second baseline,
- perimeter of the first loop (in pixels),
- angle of the moment axis of the first loop,
- perimeter of the second loop (in pixels), and
- angle of the moment axis of the second loop.

The number of baselines and loops taken into consideration was limited to two in each case. This limitation allowed us to compare the test results to our theoretical estimations, which assume a fixed number of feature properties.

The experimental setup takes \( n \) randomly chosen original signatures and \( k \) features from each signer. Afterwards, the classifier is tested with 10 more original and 10 forged signatures. The average error rate (AER) is then calculated for the whole database by averaging the error rates for each signer. The whole process is repeated 20 times (for different samples), and the results are averaged. To account for the differences of features, we evaluated the system for all possible feature combinations (all possible feature pairs, triplets, etc.) and again averaged the results. The final error rates are summarized in Table 3.4 and Table 3.5.

To allow comparisons, each table also shows the predicted error rates (b). It can be observed that the obtained results almost always fall behind the predicted ones. This is normal and is a result of the errors in the matching and feature extraction algorithms. Additionally, as the number of features increases, the precision of our verifier falls further and further behind the expected results. This finding is also normal and is a result of the interdependency between features. However, even with these two anomalies, there is a clear correlation between expected and achieved results.
Table 3.4 Classification results on the SVC2004 database

<table>
<thead>
<tr>
<th>reference signatures</th>
<th>number of features</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>44.6</td>
</tr>
<tr>
<td>3</td>
<td>43.2</td>
</tr>
<tr>
<td>4</td>
<td>42.7</td>
</tr>
<tr>
<td>5</td>
<td>42.2</td>
</tr>
<tr>
<td>6</td>
<td>41.9</td>
</tr>
<tr>
<td>7</td>
<td>41.9</td>
</tr>
<tr>
<td>8</td>
<td>41.5</td>
</tr>
<tr>
<td>9</td>
<td>41.3</td>
</tr>
<tr>
<td>10</td>
<td>41.3</td>
</tr>
</tbody>
</table>

b) Predicted error rate (%)

<table>
<thead>
<tr>
<th>reference signatures</th>
<th>number of features</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>42.5</td>
</tr>
<tr>
<td>3</td>
<td>41.1</td>
</tr>
<tr>
<td>4</td>
<td>40.4</td>
</tr>
<tr>
<td>5</td>
<td>40.1</td>
</tr>
<tr>
<td>6</td>
<td>39.8</td>
</tr>
<tr>
<td>7</td>
<td>39.7</td>
</tr>
<tr>
<td>8</td>
<td>39.6</td>
</tr>
<tr>
<td>9</td>
<td>39.5</td>
</tr>
<tr>
<td>10</td>
<td>39.4</td>
</tr>
</tbody>
</table>

c) Differences between a) and b)

<table>
<thead>
<tr>
<th>reference signatures</th>
<th>number of features</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2.1</td>
</tr>
<tr>
<td>3</td>
<td>2.1</td>
</tr>
<tr>
<td>4</td>
<td>2.3</td>
</tr>
<tr>
<td>5</td>
<td>2.1</td>
</tr>
<tr>
<td>6</td>
<td>2.1</td>
</tr>
<tr>
<td>7</td>
<td>2.2</td>
</tr>
<tr>
<td>8</td>
<td>1.9</td>
</tr>
<tr>
<td>9</td>
<td>1.8</td>
</tr>
<tr>
<td>10</td>
<td>1.9</td>
</tr>
</tbody>
</table>
Table 3.5 Classification results on the GPDS300 database

a) Achieved error rate (%)

<table>
<thead>
<tr>
<th>reference signatures</th>
<th>number of features</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>44.8</td>
</tr>
<tr>
<td>3</td>
<td>43.6</td>
</tr>
<tr>
<td>4</td>
<td>43.2</td>
</tr>
<tr>
<td>5</td>
<td>43.0</td>
</tr>
<tr>
<td>6</td>
<td>42.7</td>
</tr>
<tr>
<td>7</td>
<td>42.7</td>
</tr>
<tr>
<td>8</td>
<td>42.5</td>
</tr>
<tr>
<td>9</td>
<td>42.3</td>
</tr>
<tr>
<td>10</td>
<td>42.1</td>
</tr>
</tbody>
</table>

b) Predicted error rate (%)

<table>
<thead>
<tr>
<th>reference signatures</th>
<th>number of features</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>44.6</td>
</tr>
<tr>
<td>3</td>
<td>43.6</td>
</tr>
<tr>
<td>4</td>
<td>43.1</td>
</tr>
<tr>
<td>5</td>
<td>42.8</td>
</tr>
<tr>
<td>6</td>
<td>42.7</td>
</tr>
<tr>
<td>7</td>
<td>42.6</td>
</tr>
<tr>
<td>8</td>
<td>42.5</td>
</tr>
<tr>
<td>9</td>
<td>42.4</td>
</tr>
<tr>
<td>10</td>
<td>42.4</td>
</tr>
</tbody>
</table>

c) Differences between a) and b)

<table>
<thead>
<tr>
<th>reference signatures</th>
<th>number of features</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0.1</td>
</tr>
<tr>
<td>3</td>
<td>0.1</td>
</tr>
<tr>
<td>4</td>
<td>0.1</td>
</tr>
<tr>
<td>5</td>
<td>0.1</td>
</tr>
<tr>
<td>6</td>
<td>0.0</td>
</tr>
<tr>
<td>7</td>
<td>0.1</td>
</tr>
<tr>
<td>8</td>
<td>0.0</td>
</tr>
<tr>
<td>9</td>
<td>-0.1</td>
</tr>
<tr>
<td>10</td>
<td>-0.2</td>
</tr>
</tbody>
</table>

### 3.9 Chapter summary

An algorithmic background for feature-based off-line signature verification has been contributed in this chapter. It was shown that the distribution of several feature values can be reasonably well approximated with normal distribution. The intuitive explanation was confirmed by our experimental results. The
statement has been also extended to forged feature values, which follow the same distribution but with (usually) greater variance.

Using the previously formulated assumptions, it became possible to describe the total error rate of a threshold classifier working on a single feature (Proposition 3.1). In Proposition 3.2 we have shown that it is possible to select a threshold, which also minimizes the total error rate. As signature verification usually has to deal with a low number of reference samples, the sample count has been also incorporated in our calculations by using Student’s t-distribution to approximate the distribution of feature values (Proposition 3.3).

Next, we extended our model to take the number of features into consideration. Provided that each of the features has a binary vote about the origin of a signature, we have shown the minimal number of features that can be classified as forged before the whole signature should be classified as a forgery (Proposition 3.4) and calculated the resulting error rate of a system (Proposition 3.5).

To test the validity of our model, a complex experiment was introduced in Section 3.8.3. The experiment has shown that our analytical estimations were in good accordance with the measured results, although – because of the interdependency of feature values – these results tend to diverge when working with 10 or more features.

The main benefit of the introduced model is that – unlike the artificial intelligence-based approaches – it can be analytically described, providing a deep insight into the decision mechanisms. To demonstrate this, we are going to examine 3 major properties of our model in the next sections.
4. The implications of the statistical model

Based on the results of the previous chapter, we are now able to predict the accuracy of a signature verification system based on only 3 parameters: the number of original signatures used to train the system \( (n) \), the number of independent features \( (k) \) and the quality of the forged features \( (q) \). Moreover, the first two can be directly controlled by the experimental setup. Equation 3.42 allows us to model the effects of any of the parameters on each other. This yields several practical applications, some of which are shown in the next subsections.

4.1 The effect of feature count \( (k) \) on AER.

In the next proposition, we are going to prove that increasing the number of features will, almost always decrease and never increase the average error rate. Although this statement may seem to be a trivial consequence of our previous decisions, the fact, that this property of the verifier may be analytically proven, makes our approach unique, among the several AI based approaches. Also, some of our previous transformations (like taking the lower bound of \( l \) in Equation 3.39) do not necessarily guarantee, to keep this property for the final model.

**Proposition 4.1**

Given the constraints

\[
q > 1, k \geq 1, n > 2, n, k \in \mathbb{Z}, q \in \mathbb{R}
\]  \hspace{1cm} (4.1)

and provided that the number of reference samples \( (n_0) \) and the forgery quality \( (q_0) \) is given, the function \( AER(q_0, n_0, k) \) is decreasing.

**Proof:**

We have to prove that

\[
AER(q_0, n_0, k) \geq AER(q_0, n_0, k + 1)
\]  \hspace{1cm} (4.2)

As the parameters \( q_0 \) and \( n_0 \) are fixed, they will be omitted during this proof. As \( \alpha \) and \( \beta \) are only dependent on \( n \) and \( q \) they can be regarded as fixed numbers.

We should note some of the main properties of the arguments, which will be used in the following proofs

\[
0 < \alpha < 1
\]  \hspace{1cm} (4.3)

\[
0 < \beta < 1
\]  \hspace{1cm} (4.4)

\[
\alpha < \beta
\]  \hspace{1cm} (4.5)

\[
\frac{\alpha + \beta}{2} < 0.5
\]  \hspace{1cm} (4.6)
\[ 1 < \frac{1 - \beta}{\alpha} \quad (4.7) \]

\[ 1 < \frac{1 - \alpha}{\beta} \quad (4.8) \]

Equations 4.3, 4.4 and 4.5 are trivial, equation 4.6 is a direct consequence of 3.10 and equations 4.7 and 4.8 are transformations of equation 4.6.

**Lemma 4.1**

If

\[ l = k \log_{(1 - \alpha)(1 - \beta)} \left( \frac{1 - \alpha}{\alpha \beta} \right) \]

then

\[ \left\lfloor k \log_{(1 - \alpha)(1 - \beta)} \left( \frac{1 - \alpha}{\alpha \beta} \right) \right\rfloor = |l| \quad (4.10) \]

and

\[ \left\lfloor (k + 1) \log_{(1 - \alpha)(1 - \beta)} \left( \frac{1 - \alpha}{\alpha \beta} \right) \right\rfloor = \begin{cases} |l| & \text{if } l \leq 0 \\ |l| + 1 & \text{if } l > 0 \end{cases} \quad (4.11) \]

**Proof:**

Equation 4.10 is direct consequence of equation 4.9. To prove 4.11, we have to prove, that

\[ \log_{(1 - \alpha)(1 - \beta)} \left( \frac{1 - \alpha}{\alpha \beta} \right) < 1 \quad (4.12) \]

This is always true as both the argument (equation 4.8) and the base (equations 4.7 and 4.8) are greater than 1 and the base is greater than the argument, and all of them are positive.

Because of Lemma 4.1, the proof must be divided into two separate cases.

**4.1.1 Case 1**

\[ \left\lfloor (k + 1) \log_{(1 - \alpha)(1 - \beta)} \left( \frac{1 - \alpha}{\alpha \beta} \right) \right\rfloor = |l| \quad (4.13) \]

In this case

\[ AER(k) = \frac{FAR(k) + FRR(k)}{2} \quad (4.14) \]

\[ AER(k + 1) = \frac{FAR(k+1) + FRR(k+1)}{2} \quad (4.15) \]

We are going to prove that
After expanding and performing the summations (see Appendix A.6 for details)

\[
\frac{1 - \beta}{\alpha} \left( \frac{1 - \beta}{\alpha} \right)^{|l|} \left( \frac{1 - \alpha}{\beta} \right)^{|l|} > \frac{(1 - \alpha)^k}{\beta^k}
\]

From equation 3.37

\[
\left( \frac{1 - \beta}{\alpha} \right)^{|l|} \left( \frac{1 - \beta}{\alpha} \right)^{|l|} \left( \frac{1 - \alpha}{\beta} \right)^{|l|} \left( \frac{1 - \alpha}{\beta} \right)^{|l|} = \frac{(1 - \alpha)^k}{\beta^k}
\]

Combining the previous equations, we get:

\[
\frac{1 - \beta}{\alpha} > \left( \frac{1 - \beta}{\alpha} \right)^{|l|} \left( \frac{1 - \alpha}{\beta} \right)^{|l|}
\]

We also know from equation 4.13 that

\[
|l| + \log_{\frac{(1 - \alpha)(1 - \beta)}{\alpha \beta}} \frac{1 - \alpha}{\beta} < 1
\]

therefore

\[
\left( \frac{1 - \beta}{\alpha} \right)^{|l|} + \log_{\frac{(1 - \alpha)(1 - \beta)}{\alpha \beta}} \frac{1 - \alpha}{\beta} \left( \frac{1 - \alpha}{\beta} \right)^{|l|} + \log_{\frac{(1 - \alpha)(1 - \beta)}{\alpha \beta}} \frac{1 - \alpha}{\beta} \left( \frac{1 - \alpha}{\beta} \right)^{|l|} < \left( \frac{1 - \beta}{\alpha} \right)^{|l|} \left( \frac{1 - \alpha}{\beta} \right)^{|l|}
\]

from which

\[
\frac{1 - \beta}{\alpha} > \left( \frac{1 - \beta}{\alpha} \right)^{|l|} \left( \frac{1 - \alpha}{\beta} \right)^{|l|}
\]

The constraint ensured by equation 4.24 supports equation 4.21, which was derived from our original statement with equivalent transformations. Therefore, the original statement is true.

4.1.2 Case 2

\[
\left( k + 1 \right) \log_{\frac{(1 - \alpha)(1 - \beta)}{\alpha \beta}} \frac{1 - \alpha}{\beta} = |l| + 1
\]

The proof is mainly analogous to case 1. Only the main differences are highlighted below, while the details are provided in Appendix A.6.
We are going to prove that

\[ AER(k) \geq AER(k + 1) \]  \hspace{1cm} (4.26)

After expanding and performing the summations:

\[ \frac{1-\beta}{\alpha} \left( \frac{1-\beta}{\alpha} \right)^{[l]} \left( \frac{1-\alpha}{\beta} \right)^{[l]} \leq \frac{(1-\alpha)^{k}}{\beta^k} \] \hspace{1cm} (4.27)

Combining the above with equation 4.19 we get:

\[ \frac{1-\beta}{\alpha} \leq \left( \frac{1-\beta}{\alpha} \right)^{[l]} \left( \frac{1-\alpha}{\beta} \right)^{[l]} \] \hspace{1cm} (4.28)

We also know from equation 4.25 that

\[ \{l\} + \log \frac{(1-\alpha)(1-\beta)}{\alpha \beta} \frac{1-\alpha}{\beta} \geq 1 \] \hspace{1cm} (4.29)

therefore

\[ \left( \frac{1-\beta}{\alpha} \right)^{[l]} + \log \frac{(1-\alpha)(1-\beta)}{\alpha \beta} \frac{1-\alpha}{\beta} \geq \left( \frac{1-\beta}{\alpha} \right)^1 \left( \frac{1-\alpha}{\beta} \right)^1 \] \hspace{1cm} (4.30)

from which

\[ \frac{1-\beta}{\alpha} \leq \left( \frac{1-\beta}{\alpha} \right)^{[l]} \left( \frac{1-\alpha}{\beta} \right)^{[l]} \] \hspace{1cm} (4.31)

The constraint ensured by equation 4.31 supports equation 4.28, which was derived from our original statement with equivalent transformations, therefore the original statement is true.

Figure 4.1 provides a visual representation of the results (for n = 2).
Remark 1: Although we allowed for equality in Equation 4.2, this equality arises only in the extremely unlikely case when the left side of Equation 4.29 equals 1. Thus it is safe to say that increasing the feature count will lead to better classification results.

4.2 The effect of sample count \((n)\) on AER

In Proposition 3.3 we have shown how to choose the optimal decision threshold for a given sample count \((n)\). However, this does not guarantee, that a higher value of \(n\) will yield lower error rates.

In the next proposition we are going to show that increasing the number of reference signatures will always decrease the average error rate.

Proposition 4.2

Given the constraints

\[
q > 1, k_0 = 1, n > 2, n, k \in \mathbb{Z}, q \in \mathbb{R}
\]  

(4.32)

and provided that the number of observed features \((k_0)\) and forgery quality \((q_0)\) is given, the function \(AER(q_0, n, k_0)\) is strictly decreasing.

Lemma 4.2

As the number of degrees of freedom grows, \(t\)-distribution approaches normal distribution. Therefore, as the number of samples increases, both the distributions of original and forged samples will converge to normal distribution, resulting in a total error rate converging towards the average error of normal distributions, which was described in Equation 3.7:
\[ \lim_{n \to \infty} AER(q, n, 1) = \frac{1}{2} + \Phi \left( \frac{w}{q} \right) - \Phi (w) \]  

(4.33)

**Proof:**

Using analytical tools [67], we calculated the partial derivative \( \frac{\partial AER(q, n, 1)}{\partial n} \) (see Appendix A.7 for details). Our simulation (Figure 4.2) indicates that this derivative is negative for the relevant input range \((n \in [2; 80], q \in (1, 10])\) and it suggests that this is also the case for higher values of \(n\). To support the latter assumption, we will show that the function value converges towards zero.

Lemma 4.2 shows that AER converges to \( \frac{1}{2} + \Phi \left( \frac{w}{q} \right) - \Phi (w) \), which is not dependent on \(n\), therefore \( \frac{\partial AER(q, n, 1)}{\partial n} \) will converge towards 0.

\[ \lim_{n \to \infty} \frac{\partial AER(q, n, 1)}{\partial n} = 0 \]  

(4.34)

Since the derivative of the function has a limit of zero and based on the fact that the graph has approached it significantly from below, we can assume that it will not obtain positive values outside the plotted range Figure 4.2.

Figure 4.2 The function \( \frac{\partial AER(q, n, k)}{\partial n} \) 
(values below \(-0.001\) were removed)

Figure 4.3 illustrates that an increase of \(n\) will always result in a lower error rate.
Remark 1: The statement can be generalized for $k \geq 1$ based on Proposition 4.1.

Remark 2: The improvement in error rates decreases rapidly and is almost insignificant when $n > 10$. In conclusion: increasing the sample sizes above 10 has a negligible effect on practical application.

4.3 The effect of threshold ($w$) on AER

In the previous sections we have seen that the value of $q$ influences the accuracy of the system. Figure 4.4 shows that although this influence is important, there is usually a wide range for the threshold ($w$) where the differences only play a minor effect on the final classification results. This section is devoted to formulating a conjecture about this property and providing experimental results to verify it.
The implications of the statistical model

Figure 4.4 The relationship of $q$, $w$ and AER when $n = 10$ and $k = 10$

Figure 4.5 shows some cross-sections of Figure 4.4 at given forgery qualities ($q$). At $q = 1.01$, the AER is close to 0.5 because there is almost no way to distinguish between originals and forgeries. By increasing the value of $q$, the achievable error rate becomes lower. The graph also shows that when $q \approx 4$, the value of $w$ may fall anywhere between 2 and 3.5 without major effects on AER.

Of course, in our application, $w$ is set by our system (Equation 3.18) depending on a previously estimated value of $q$. Figure 4.6 shows the resulting increases of error rates resulting from misestimating $q$. 
The wide “plane” on Figure 4.6 shows, that for example when the actual value of $q$ is 17.5 then calculating the decision threshold based on any value of $q$ between 4.7 and 20 will result in a less than 2% growth in error rate compared to the optimal choice of $q$.

More interestingly, if we set the estimated value of $q$ to 5.5, the calculated error growth will stay below 2% independently of the actual value of $q$. This later statement is only true, because Figure 4.6 sets an upper bound (20) to both the estimated and the actual forgery qualities. Such a restriction is not unpractical, as in our experiments the actual value of $q$ was usually below 5 and – with the exception of rare extreme cases – never went over 20. In the next conjecture we are going to generalize these observations, by limiting the value of $q$ and observing the possible resulting growth of error rates.

**Conjecture 4.1**

Supposed that the value of the actual forgery quality ($q_{act}$) falls in the interval $(1; q_{max})$, there is an ideal value for estimated forgery quality ($q_{est}$) that minimizes the errors resulting from a suboptimal choice of $w$.

**Experimental results**

Figure 4.7 shows that there is a clear choice for $q$ that can minimize the total error, provided that the maximal value of $q$ is known. Figure 4.8 shows these optimal choices, while Figure 4.9 shows the resulting worst-case scenario for an increase in classification errors.
Figure 4.7 The resulting increase in error depending on the maximal value ($q_{max}$) and the estimated value ($q_{est}$) of forgery quality.

Figure 4.8 The optimal choices for $q$, depending on the maximum of $q_{act}$.
4.4 Chapter summary

In this chapter we have introduced three important characteristics of our statistical model.

In Proposition 4.1 it was shown that the average error rate of the system decreases as the number of features increases. This relation is inversely proportional to $q$. While at $q = 1.01$, this decrease is very slow, at $q = 10$ even 3 feature values are enough to lower the expected error rate to below 10%.

Proposition 4.2 has shown a similar relation between the number of reference signatures and the average error rate. Although the statements of the previous propositions may seem trivial at first glance, even the fact that they could be proven shows that our statistical classifier exposes a unique analytical potential that most other classifiers lack.

Conjecture 4.1 provides a deeper insight into our model by examining possible error increases that arise when the quality of forgeries was misestimated. Our simulations show that this kind of error can be greatly reduced by the optimal choice of parameter $q$. For example, when the actual value of $q$ is somewhere between 1 and 20 and we estimate it as $q = 5.3$, then the increase of $AER$ will stay below 2% even in the worst possible case.

Last, but not least, all of the previously introduced calculations are highly dependent on the accuracy of previous processing steps such as feature extraction and feature matching. Therefore, the next chapter is devoted to the introduction of a novel stroke extraction algorithm and the main processing algorithms used to provide the input for our classification system.
5. **Signature processing algorithms**

The previous two chapters have shown how feature values can be used in the classification phase of signature verification. In order to achieve these results, the features first had to be located and assigned to each other in various processing steps. Moreover, the accuracy of these steps highly influences the achievable error rates of the classifier.

This section describes a collection of processing algorithms used in our solution. First, a brief overview of thinning approaches is given and a new thinning algorithm is proposed for the extraction of stroke information from signature images. The remainder of the section describes the matching and feature extraction algorithms used to extract and match loops and baselines in our system.

5.1 **Line thinning**

The input of off-line signature verifiers consists of scanned images of handwritten signatures that usually cannot be processed in their original form by some modules of the verification system. The purpose of thinning in the preprocessing phase is to extract a binary, one pixel wide skeleton from the signature. Optimally, this discrete skeleton is a compact representation of the original signature that minimizes the loss of semantic information; therefore, it can be efficiently interpreted by the other modules of the process. Thinned images make several later steps – like the extraction of endpoints in Figure 5.1 – much simpler (and thereby faster) by reducing the amount of information to process.

After giving an overview of the related work and currently applied methods, we will propose a novel thinning algorithm. Experimental validation of the algorithm is presented and evaluated. This algorithm could be used to refine existing signature verification methods and also to create new ones [68].

![Figure 5.1 Endpoint extraction on a thinned image](image)

5.1.1 **The classification of thinning algorithms**

Thinning methodologies have gone through plenty of changes and improvement in the last decades. They can be categorized as follows:
5.1.1.1 Morphology-based approaches

The first attempt at extracting skeletons from binary images was using the operators of mathematical morphology (MM), widely used at the time for texture analysis.

The basic idea in binary morphology [69] is to probe an image with a simple, pre-defined shape (called the probe, kernel or structuring element), then to draw conclusions based on how this shape fits or misses the shapes in the image. For thinning, the most important operation of MM is erosion (Figure 5.2), which basically erodes the boundaries of the region in the foreground. Thus areas of foreground pixels shrink in size and holes within those areas become larger.

![Figure 5.2 Erosion of the blue square by a disk, resulting in the red square](image)

Erosion in itself does not produce appropriate skeletons, but its logic is clearly present in more complex algorithms, especially that of windowing.

5.1.1.2 Raster scanning based approaches

The execution of raster scan algorithms usually goes as follows [70]:

1. All the pixels of a binary image are examined in a predetermined order
2. A set of conditions is evaluated for each pixel, either marking it for deletion or for retention
3. At the end of the iteration, all the pixels marked for deletion are erased from the image and the process jumps back to step 1.

If no pixel was deleted in the iteration, the thinning process is done and the remaining black pixels form the skeleton of the image.

Raster scan algorithms differ from each other in the set of conditions evaluated to decide whether to delete or retain a particular pixel. Different condition sets preserve different characteristics of the input image.

---

4 In some cases, only the borders of connected objects are examined (contour following).
5.1.1.3 Windowing-based approaches

A specialized version of the hit-and-miss morphological operator is used in numerous thinning algorithms. It is usually referred to as windowing or masking.

A thinning mask is an arbitrary-sized – though usually small – grid in which each cell denotes a custom condition. The mask is fitted onto a region of the tested image, and then the conditions of the cells are evaluated for the pixel under it. We say that the mask fits onto the region of the image if every pixel under the mask satisfies the corresponding condition.

This technique is used in raster scan algorithms where the conditions evaluated for each pixel are given in the form of a thinning mask [71].

In Figure 5.3, an example of thinning masks can be seen where the meaning of the mask cells are the follows:

- Pixels marked with 0 have to belong to the background (white)
- The pixel marked “P” is the center point, the point where the mask is fitted onto an image
- Each group of pixels marked with A or B must have at least one foreground (black) pixel

The mask is used to preserve connected components of an image, because if it (or any of its 90° rotation) fits onto a pixel, it means that erasing it would break apart a connected component into two distinct parts.

```
A A A
0 P 0
B B B
```

Figure 5.3 Example of a commonly used thinning mask

5.1.1.4 Complex methods for line thinning

There are numerous other algorithms that do not fit into the above categories. Most are based on a unique idea with which the skeleton of an image can be extracted.

One example is the Sparse Pixel Vectorization method in which a ray is casted and traced inside the foreground of the image components and then the medial points of the rays are connected [72]. Another example is the line following algorithm proposed by Peng in [73], which attempts not to iteratively erode contour pixels, but to extract the skeleton by following the two sides of the stroke in the input image (this method is a natural approach, as the human eye identifies lines in the same way). Our own research also includes several alternative approaches e.g. the physical simulation of a pen [74].

These algorithms are usually more complex to implement and can be quite vulnerable to various attributes of the input skeleton, thus careful configuration is needed in each environment (depending on the line width and other properties of the input shapes). In return, they can perform reasonably well in certain areas
and usually have low computational cost since they do not have to examine every pixel, nor to iterate recursively.

5.1.1.5 Stroke extraction

In signature verification, the notion of thinning is often strongly coupled with the reconstruction of the drawing order of the thinned lines [75]. This combined approach is called stroke extraction. Several stroke extraction methods have been introduced in the past. Some robust stroke extraction solutions have been developed for the purpose of recognizing handwritten text [76] [77] and there seems to be an extensively wide study of extracting strokes from Chinese characters [78] [79] [80] [81]. However, these methods tend to work at a high level of abstraction (they focus on recognizing letters and words) and are thereby not suitable for detecting the fine features used in signature verification.

Another class of methods is based on simple line tracing. Either because the resolution of the signature is already low [45], or mostly because they apply some line thinning algorithms [82] [50] [38]. In both cases, the loss of semantically important information [11] is high. A three-stage stroke extraction involving a heuristic stroke following method has been proposed in [83]. However, this only targeted characters and graphemes.

5.1.2 The Scanning Circle Algorithm

In the following section, a robust algorithm is introduced with the purpose of identifying how exactly the signer wrote his signature. The main goal was to create an algorithm that performs well on noisy, unprocessed images.

In general, this method traces a signature using the image of it, extracts control points, determines their order, and finally assigns them to strokes. This produces a graph representation of the signature, which can be used for further processing. As several steps in the algorithm require the scanning of pixels around a circle, we are going to refer to it as the Scanning Circle Algorithm (SCA).

First, the core of the algorithm is introduced, then this is refined to cope with some specific challenges.

5.1.2.1 Basic stroke extraction algorithm

The following basic concepts and variables are used in the algorithm:

- **Signature point**: a pixel in the image that belongs to the signature
- **Stroke point**: the control points of the polylines representing the strokes
- **p**: average pen width (in pixels)
- **r**: the radius of the scanning circle (in pixels) initially \( r = 2p \)
- **d**: threshold (in pixels), where \( d < r \) initially \( d = r - 1 \)
- **Weighted middle point**: barycenter. Given a series of points \( P_1, P_2 ... P_n \) with corresponding intensity values of \( i_1, i_2 ... i_n \), the weighted middle point \( P_m \) is the first point in the series where
\[
\sum_{k=1}^{m} i_k > \frac{1}{2} \sum_{k=1}^{n} i_k
\]
(5.1)

- **Free point**: a pixel which has a minimum distance of \( d \) from a previously detected stroke point
- **Connected points**: pixels which can be connected with a straight path consisting only of signature points

Using the above definitions, the algorithm is defined as follows:

**Step 1 – Locate a signature point**

1. Going from the bottom to the top and from the left to the right, locate the first free signature point \((P)\). This point is going to be the starting point of our next stroke: \( S := P \).
2. If there is no free point, then exit.

**Step 2 – Find connected stroke points (see Figure 5.4)**

3. Examine all points of a circle with the center \( S \) and the radius \( r \). More than \( p/2 \) adjacent signature points on the circle define arcs. Select the weighted middle points \((A_1, A_2 \ldots A_n)\) of these arcs as possible stroke points.
4. Deselect \( A_i \) where \( A_i \) is not free, or not connected with \( S \). (see Figure 5.4)

![Figure 5.4 Step 2 of the stroke extraction algorithm](image)
Step 3 – Finish the stroke (if needed) and repeat

5. If count(selected \( A \)) is

0: Finish the current stroke and go to Step 1

1: Take the arc point as the next stroke point \( (S := A) \) and go to Step 2

> 1: Finish the stroke and begin two (or more) new strokes with starting point \( S \), and take \( A_1, A_2 \ldots \) as the second point. Now, follow Step 2 for each of them.

Figure 5.5 shows this logic in a form of a flow chart. A sample run of the algorithm is illustrated in Figure 5.6.
5.1.2.2 Parameterization and extensions

In the field of pattern recognition, the usage of fixed thresholds and parameters is usually avoided \([84]\). Therefore, the parameters \((p, r, d)\) used in the algorithm are all defined as functions of the image contents \((p)\) or of each other \((d, r)\), thereby allowing the algorithm to be applied to different pen types and different resolutions without manual adjustments.

There are still several elements that can strongly affect the performance of the algorithm. The definition of the “signature point” deliberately does not state how to determine whether a pixel is part of the signature or not. The simplest way of doing this would be to perform noise filtering and a binarization of the original image, perhaps adding morphological closing, as well. This may be adequate for handwriting recognition. However, in the field of signature verification, every single pixel value can be of great importance, therefore a more sophisticated implementation was chosen. Instead of binarization, the pixels of the image are divided into three classes (ink, paper and undefined), using two intensity thresholds.

Every pixel which has an intensity below the lower threshold is assigned to the “ink” class. These pixels probably belong to the signatures. Every pixel with intensity values above the higher threshold is assigned to the “paper” class. All other pixels (with intensity values between the lower threshold and the upper threshold) are assigned to a new class called “undefined”.

In Step 1 a pixel is considered to be a “signature point” if in a radius of \(p/2\) pixels around it less than 20% of the pixels belong to the paper class. This limit was found to yield the best results. However, this could vary depending on noise ratio and ink type.

To improve performance in step 2, only pixels on the circle around \(S\) are considered, and even a single pixel belonging to the “paper” class can break a row of adjacent pixels. In both cases at least 50% of the pixels must belong to the “ink” class.
The definition of connected points also needs some adjustment. Connected points are pixels that can be connected with a straight path consisting only of ink and undefined points where the number of connected undefined points cannot exceed \( p/4 \), and the total number of undefined points must be less than the total number of ink points.

### 5.1.2.3 Stroke sequence estimation

When reaching junction points, several new strokes are defined. This is a necessary side effect of the algorithm which is easy to eliminate in a post processing step. The aim of the stroke sequence estimation is to combine these strokes in a way that will make the resulting strokes correspond to the original handwritten strokes. In Figure 5.7, this would mean that \( a_1 \) and \( a_2 \) (and similarly \( b_1 \) and \( b_2 \)) have to be combined into a single stroke.

![Figure 5.7](image)

Figure 5.7 The stroke extraction algorithm detects 4 different strokes \((a_1, a_2, b_1, b_2)\) in the image, with one common point.

After the stroke extraction algorithm, connected strokes will have common endpoints – because of step 3 –, therefore these common endpoints can be used to locate junctions. A junction point is the meeting point of at least 3, but possibly more strokes.

There are several existing solutions for stroke sequence estimation in the literature. In [85], a mainly local approach is taken by examining typical patterns of thinned images. [86] and [87] take a more generic approach by encapsulating local and global heuristics in the algorithm. As [87] states, to make optimal decisions, almost every combination of strokes must be examined which yields to a combinatorial explosion. To avoid this, our approach will be limited to local decisions (only one junction is considered for examination at a time). This also allows us make the best possible local choice.

We defined the stroke sequence estimation algorithm as follows: Let \( n \) be the count of strokes starting from a common junction point. It is assumed that at least \( \lceil n/2 \rceil \) stroke pairs have to be combined, therefore all possible stroke-pair combinations are examined. A distance function which calculates the differences of tangents at the junction points is applied to each stroke pair to calculate the goodness of the match. The best combination is where the sum of distances is the lowest.
A sample run of the improved algorithm is demonstrated in Figure 5.8.

Figure 5.8 A sample run of the improved stroke extraction algorithm
The number of strokes was reduced from 22 to 13.

5.1.2.4 Adaptive circle radius
Although the original algorithm performs well when following straight lines, curves and even when detecting junctions, the high value of the scanning radius \( r \) makes it impossible to correctly detect sharp curves in the original signature’s strokes. This can result in the detection of phantom strokes (Figure 5.8).

To tackle this problem, Step 2 is modified to allow the changing of \( r \) (Figure 5.9). When the angle between two stroke segments falls below a given angle or the number of points in an arc goes significantly beyond the pen width \( p \), a new scanning circle is tested with a reduced radius. The radius is decreased several times if necessary, but it cannot get under \( p/2 \). In the next iteration, Step 2 will start with the reduced radius, but contrary to the previous case, it will try to increase the radius if it can whilst maintaining a maximum value of \( 2p \). Figure 5.10 illustrates the improved results.

Figure 5.9 Stroke extraction
with constant (a) and with adaptive (b) scanning radius.
5.1.2.5 Experimental results

Because of the use of several heuristic methods in the algorithm, a continuous monitoring of accuracy is essential. Although we were able to validate single cases with human interaction, a statistical validation is hard to obtain because of the lack of on-line information. To compensate for this, the SVC2004 database was used. This corpus was constructed from an on-line signature database, therefore it contains the original stroke information to be compared to the extracted values.

A comparison can be given by comparing the actual number of strokes to the stroke count detected by SCA. Figure 5.12 compares these average stroke counts for each of the signers in the SVC2004 database. It can be seen that our algorithm always overestimates the count of strokes, which can be tracked back to the fact that nearby strokes do not always have a common point. Therefore the connection of nearby strokes is not always possible (Figure 5.11).

However, these results are still promising, because these “extra” strokes are detected in the same way in all signatures. Figure 5.12 shows the standard deviation of the detected stroke counts for each of the signers. The average value of 1.3 is relatively low given that there are signatures with more than 14 detected strokes. Our experiments also show, that additional strokes are usually
detected at the same places in each of the signer’s signatures, thereby still allowing a clear matching in later verification phases.

Figure 5.12 Average and standard deviation of stroke counts

Figure 5.13 shows that the changes of the previous three sections really improved the accuracy of the stroke extraction algorithm by reducing the standard deviation of the detected stroke counts.

Figure 5.13 Histogram of stroke count deviations in the original and improved SCA implementation
5.1.3 Comparison with other algorithms
There is no widely adopted strategy, nor a common corpus for comparing stroke extraction algorithms; therefore we are going to limit our experiments to the evaluation of the thinning capabilities of SCA.

The process uses reference images of the SVC2004 corpus. The thinned result of each algorithm is compared to the ideal reference skeleton, which is obtained from the on-line signature information (Figure 5.14). The starting point of the comparing technique is the evaluation algorithm introduced by Pratt in [34].

![Reference skeleton and the reconstructed signature](image)

Figure 5.14 Reference skeleton and the reconstructed signature

5.1.3.1 Pratt Evaluation
Let $N_R$ and $N_A$ be the number of foreground (black) points in the reference skeleton and the actual thinned result, respectively. The distance between the $i^{th}$ pixel of the result skeleton and the pixel of the reference skeleton nearest to it is denoted by $d_i$. The Pratt evaluation score is defined by

$$PES = \frac{1}{N_M} \sum_{i=1}^{N_A} \frac{1}{1 + a_P d_i^2}$$

(5.2)

where $N_M = \max\{N_R, N_A\}$ and $a_P$ is a scaling constant, adjusting the level of penalization for distances.

![Graphs](image)

Figure 5.15, the evaluation score is calculated as follows:

$$N_R = 5, N_A = 4$$

(5.3)

$$d_1 = 1, d_2 = 1, d_3 = 0, d_4 = 0$$

(5.4)

$$PES = \frac{1}{5} \left( \frac{1}{1 + a_P d_1^2} + \frac{1}{1 + a_P d_2^2} + 1 + 1 \right) = 0.76$$

(5.5)

if $a_P$ is chosen to be $1/9$ (which is a typically used value [34]).

If the two compared skeletons are identical to each other, then the result is 1.0 while lower values mean a lower level of similarity.
5.1.4 The experimental setup

The following fourteen algorithms have been implemented [88] and tested with the above evaluation method: raster scan by Rutovitz, 1966 [71], raster scan by Yokoi, 1973 [70], window matching by Beun, 1973 [70], contour scan by Arcelli, 1978 [70], window matching by Pavlidis, 1981 [70], raster scan by Holt, Stewart, Clint and Perrott, 1987 [71], window matching by Chin, Wan, Stover and Iverson, 1987 [71], raster scan by Zhang and Wang, 1988 [71], raster scan by Eckhardt and Maderlechner, 1993 [71], window matching by Wu and Tsai, 1992 [71], raster scan by Guo and Hall, 1992 [71] (see Figure 5.17 for an example), window matching by Jang and Chin, 1992 [70], raster scan by Hilditch, [70], and the Sparse Pixel Vectorization method by Dov Dori and Wenyin Liu, 1999 [72].

The first signature from each of the signers of the SVC2004 database was used to evaluate the different algorithms (Figure 5.16). The average score of the algorithms is summarized in Table 5.1. It can be seen that the best algorithm has a Pratt-score of 0.881, while SCA is only 2.3% behind with 0.858.
In order for the thinned images to be useful in the reconstruction of strokes, they must preserve connectivity. Several of the examined methods do not satisfy this criterion, which is not desirable; therefore we have examined this property of each of the introduced algorithms. Finally, as a third aspect of the analysis we have added the average ratios of pixels in the extracted and in the reference signatures.

Experimental results show that the accuracy of SCA is comparable to the best thinning algorithms examined. This makes the algorithm a suitable base for further use in a signature verification process, which is the subject of our ongoing research.
Table 5.1 The average ratings, tested with 20 signatures

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Pratt rating</th>
<th>Keeps connectivity</th>
<th>Extracted/reference pixels</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSCPRasterScan</td>
<td>0.881</td>
<td>yes</td>
<td>1.050</td>
</tr>
<tr>
<td>JangChinWindowMatching</td>
<td>0.877</td>
<td>yes</td>
<td>0.936</td>
</tr>
<tr>
<td>RutovitzRasterScan</td>
<td>0.876</td>
<td>yes</td>
<td>1.114</td>
</tr>
<tr>
<td>ZWRasterScan</td>
<td>0.867</td>
<td>yes</td>
<td>1.049</td>
</tr>
<tr>
<td>WTWindowMatching</td>
<td>0.867</td>
<td>yes</td>
<td>0.946</td>
</tr>
<tr>
<td>YokoiRasterScan</td>
<td>0.860</td>
<td>yes</td>
<td>0.933</td>
</tr>
<tr>
<td>SCA</td>
<td>0.858</td>
<td>yes</td>
<td>0.942</td>
</tr>
<tr>
<td>GuoHallARasterScan</td>
<td>0.849</td>
<td>no</td>
<td>1.063</td>
</tr>
<tr>
<td>HilditchRasterScan</td>
<td>0.843</td>
<td>yes</td>
<td>0.929</td>
</tr>
<tr>
<td>BeunWindowMatching</td>
<td>0.836</td>
<td>yes</td>
<td>0.930</td>
</tr>
<tr>
<td>EckhardtMRasterScan</td>
<td>0.821</td>
<td>no</td>
<td>1.349</td>
</tr>
<tr>
<td>PavlidisWindowMatching</td>
<td>0.818</td>
<td>yes</td>
<td>1.413</td>
</tr>
<tr>
<td>GuoHallBRasterScan</td>
<td>0.809</td>
<td>no</td>
<td>0.945</td>
</tr>
<tr>
<td>SPVThinning</td>
<td>0.768</td>
<td>no</td>
<td>0.872</td>
</tr>
<tr>
<td>ArcelliCountourScan</td>
<td>0.576</td>
<td>yes</td>
<td>0.960</td>
</tr>
<tr>
<td>CWSIWindowMatching</td>
<td>0.556</td>
<td>no</td>
<td>0.653</td>
</tr>
</tbody>
</table>

5.1.5 Additional benefits of the SCA

While the previous comparison was limited to the overall comparison of pixel positions, we would like to emphasize some additional useful characteristics of our algorithm:

- **Dynamic extraction of control points**: the output of our algorithm consists only of a limited set of control points (instead of a full image). The distance of these control points is chosen adaptively, to efficiently represent both smaller and larger changes in stroke direction.

- **Location of junctions**: several morphologic approaches tend to have difficulties in locating the exact middle point of a junction. Our adaptive circle based approach provides special heuristics to handle the junction of two strokes and to locate their central point.

- **Taking advantage of intensity values**: contrary to binary morphology, our approach can take advantage of intensity information, to improve the localization of the skeleton. This aspect has been proven helpful, when analyzing noisy images, or compensating for bad pen quality.

5.2 Feature matching

In this section, the algorithms used to match the features of the signatures will be discussed.

5.2.1 Related work

"Image registration is the process of transforming different sets of data into one coordinate system. Data may be multiple photographs, data from different
sensors, from different times, or from different viewpoints” [89]. A large number of different applications use image registration, from medical imaging to remotely sensed data processing. There are two major categories of image registration [90], [91] area-based and feature-based methods.

Area-based methods are correlation-like techniques. These algorithms use Fourier properties and other means of structural analysis. There are some limitations, e.g. these methods often work with a rectangular window and if the images are deformed by more complex transformation, the window is not able to find some parts of the scene. Since there can be large variability between the signatures of a signer, area-based methods are less applicable in the case of off-line signature analysis.

Feature-based methods assume that two sets of features have been detected. These find the correspondence between reference and examined images using their spatial relations or various feature descriptors: lines, loops (Figure 5.18), curves, points, line intersections, boundaries etc.

One innovative way of doing this is the one Valyon et al. use in [92] as preprocessing, a hybrid approach creating a point-to-point matching for further decision-making.

To allow a comparison of features, a mapping should be defined between the features of each signature pair. This happens in two phases. During the training phase, a mapping should be established among the features of the reference (original) signatures. During testing, the features of a sample signature should be mapped to the previously matched features. This is a multidimensional assignment problem [93] which is NP hard; therefore we propose special heuristics for it, to take some special properties of the signatures into consideration.

5.2.2 Matching reference signatures
The graph in which the multidimensional assignment problem arises can be defined as follows.

Definition 5.1
- let the nodes of the graph represent the features in the signatures
let the edges of the graph represent the similarity of features. Therefore, an edge should be defined between each node-pair except the node-pairs within the same signature. The weights of the edges represent how good a matching is (calculated from shape similarity and Euclidean distance).

During the matching process, all original samples of the signer are processed together. We focus on identifying matches between features which are available in most of the signatures (a threshold of 80% was used in our experiments). Signatures which do not have a given feature will be ignored when calculating the statistics for it. This is a necessary tradeoff to compensate for the diversity of original signatures and errors in previous processing steps.

The matching process consists of the following steps:

*Step 1 – Pairwise matching of signatures (Figure 5.19)*

1. Take the nodes representing a signature pair and the edges between them. Calculate the optimal matching between them using the Hungarian algorithm [94] and remove all other edges.
2. Do this for each signature pair.

*Step 2 – Enumerating subgraphs (Figure 5.20)*

3. Take a node from the graph. This node is connected with at most 1 node from each other signature. Count the edges of this subgraph.
4. Do this for each node in the graph.

*Step 3 – Removing the best match (Figure 5.21, Figure 5.22, Figure 5.23)*

5. Take the subgraph from step 2 with the highest edge count.
6. Let the nodes from this graph define the mapping of a common feature.
7. Remove the nodes and the edges of this subgraph from the original graph.
8. Repeat from step 2 to locate further feature mappings.

We found that the above algorithm produced practical mappings between features in a time-efficient manner. The main steps of the algorithm are illustrated in the following figures.
Figure 5.19 The pairwise matching of signatures: letters represent features, while lower indexes identify the signature

Figure 5.20 The combined graph of feature pairs

Figure 5.21 Iteration 1 of step 3
5.2.3 Matching sample signatures

The next algorithm allows us to map the features of a new signature to the already mapped common features. For this, we also use the previously defined graph representation, extending it with the nodes of the new sample.

Step 1 – Pairwise matching of the sample to the reference signatures

1. Take the nodes from a reference signature and the nodes from the sample signature and the edges between them. Calculate the optimal matching between them using the Hungarian algorithm [94] and remove all other edges.
2. Do this for each reference signature.

Step 2 – Finding the best match (Figure 5.24)

3. Assign each node of the sample signature to the common feature to which it has the most connections.
5.3 Feature models and extraction algorithms

This section provides a brief description of the properties and extraction of features used in our previous experiments.

5.3.1 Global features

The bounding rectangle is defined as the smallest rectangle needed to enclose the signature. As the bounding rectangle may be severely influenced by a single misplaced stroke or accent, a special variant is also introduced. The trimmed bounding rectangle is the smallest rectangle needed to enclose the binarized version of the signature image by ignoring the first 5% of pixels from each of the four directions (top, bottom, left and right). Based on these rectangles the following global features were used to characterize a signature:

- **Width**: the width of the bounding rectangle
- **Height**: the height of the bounding rectangle
- **Aspect ratio**: width / height
- **Trimmed width**: the width of the trimmed bounding rectangle
- **Trimmed height**: the height of the trimmed bounding rectangle
- **Trimmed aspect ratio**: trimmed width / trimmed height

5.3.2 Baseline model

The ASTM International standard defines a baseline as “the ruled or imaginary line upon which writing or typewriting appears to rest” [95]. Baseline information consists of a set of straight lines (Figure 5.25). Each line represents an imaginary foundation of a component (usually a word), which can be regarded as an autonomous element of the signature [96]. This definition also allows us to assign baseline information to the gaps between signature elements (according to [11] those gaps are just as characteristic as any other feature of a signature). The following 6 parameters were used to describe a baseline:

- horizontal start position (X),
- vertical start position (Y),
- horizontal end position (X),
- vertical end position (Y),
• length,
• angle.

![Signature Image]

Figure 5.25 Baselines obtained by our algorithm

The extraction of baselines is performed in a multi-step image-processing algorithm illustrated in Figure 5.26. Next, the main steps of the algorithm are described. As the algorithm was implemented using the image processing components of the AForge.NET Framework [97], the corresponding AForge components are shown in brackets (when applicable).

1. Take original signature
2. Convert to grayscale, using R-Y algorithm (Intensity is calculated from color components as: \( y = \text{red} \times 0.5 + \text{green} \times 0.419 + \text{blue} \times 0.081 \)).
   
3. Binarize using iterative threshold algorithm [98]
   
   [AForge.Imaging.Filters.IterativeThreshold]
4. Invert image
   
   [AForge.Imaging.Filters.Invert]
5. Apply dilatation operator twice with a 3x3 square structuring element
   
   [AForge.Imaging.Filters.Dilatation]
6. Apply erosion operator twice with a 3x3 square structuring element
   
   [AForge.Imaging.Filters.Erosion]
7. Identify connected components
   
   [AForge.Imaging.Filters.ConnectedComponentsLabeling]
8. Extract the lowest pixel for each column of each component (lower envelope) and remove overlapping components
9. The baselines are calculated from the pixels of the lower envelopes through linear regression
5.3.3 Loop model

A loop is a “formation that curves and crosses itself” [95]. Loops – in our interpretation – are connected regions in the image that are fully enclosed by the pixels of the signature (Figure 5.27).

This definition implies the following 3 important properties:

- Pixels should be unambiguously classified as either belonging to the background (“paper” pixels) or belonging to the signature (“signature” pixels). This is currently done by testing the color components of a pixel against some fixed thresholds.
- The region must be connected. Although it sounds logical at first glance, this is against the traditional definition of a loop, which can be interrupted by other lines. However, this simplification allows a much faster processing of the image.
- Using fully enclosed regions showed to be an unrealistic target. Because of flaws of the pen and errors of the scanning process, there are often 1-2 pixel wide interruptions in pen strokes, which would break our definition of loops. To eliminate them, a morphological closing [99] is applied (two dilations and two erosions with a 3x3 structuring element) to each image before loop extraction.

Shape descriptors [84] are used to describe the different aspects of loops, thereby allowing an easy comparison in later phases [100]. There are several promising formulas described in the literature for calculating shape descriptor values. Instead of choosing one of them, we used as many of them as possible. This will allow us to identify the most significant shape factors in later phases.

The following shape descriptors were used during feature extraction: perimeter, area, formfactor, maximum diameter, maximum diameter angle, roundness,
centroid, bounding box, inscribed diameter, extent, modification ratio, compactness, bounding circle, moment axis angle, convexity, solidity, aspect ratio.

Nontrivial descriptors are defined as follows:

\[ Modification\ Ratio = \frac{\text{Inscribed Diameter}}{\text{Maximum Diameter}} \]  
\[ (5.6) \]

\[ Formfactor = \frac{4\pi \cdot \text{Area}}{\text{Perimeter}^2} \]  
\[ (5.7) \]

\[ Extent = \frac{\text{Area}}{\text{Bounding Rectangle Area}} \]  
\[ (5.8) \]

\[ Compactness = \sqrt{\frac{\frac{1}{4} \cdot \text{Area}}{\text{Maximum Diameter}}} \]  
\[ (5.9) \]

\[ Convexity = \frac{\text{Convex Perimeter}}{\text{Perimeter}} \]  
\[ (5.10) \]

\[ Solidity = \frac{\text{Area}}{\text{Convex Area}} \]  
\[ (5.11) \]

\[ Aspect\ Ratio = \frac{\text{Maximum Diameter}}{\text{Minimum Diameter}} \]  
\[ (5.12) \]

A description of shape descriptors can be found in [84].

The extraction of loops is performed in a multi-step image-processing algorithm illustrated in Figure 5.28. The main steps of the algorithm are the following (with the corresponding AForge.NET Framework [97] components in brackets):

1. Take original signature
2. Convert to grayscale, using R-Y algorithm (Intensity is calculated from color components as: \( y = \text{red} \cdot 0.5 + \text{green} \cdot 0.419 + \text{blue} \cdot 0.081 \).  
   \[ \text{Grayscale.CommonAlgorithms.RMY} \]
3. Binarize using iterative threshold algorithm [98]  
   \[ \text{AForge.Imaging.Filters.IterativeThreshold} \]
4. (1st Copy) Flood fill (4 directional) with black color starting from top first pixel \[ \text{AForge.Imaging.Filters.PointedColorFloodFill} \]
5. (1st Copy) Fill black holes in white objects  
   \[ \text{AForge.Imaging.Filters.FillHoles} \]
6. (2nd Copy) Invert image
7. (2nd Copy) Apply dilatation operator twice with a 3x3 square structuring element \[ \text{AForge.Imaging.Filters.Dilatation} \]
8. (2nd Copy) Apply erosion operator twice with a 3x3 square structuring element \([AForge.Imaging.Filters.Erosion]\)

9. (2nd Copy) Invert image
   
   (2nd Copy) Flood fill (4 directional) with black color starting from top first pixel \([AForge.Imaging.Filters.PointedColorFloodFill]\)

10. (2nd Copy) Fill black holes in white objects
    \([AForge.Imaging.Filters.FillHoles]\)

11. Add 1st Copy and 2nd Copy

12. Remove objects smaller than 5x5 pixels
    \([AForge.Imaging.Filters.BlobsFiltering]\)

13. Label connected components with different colors
    \([AForge.Imaging.Filters.ConnectedComponentsLabeling]\)

14. Project components on the original signature (for monitoring)

15. Extract loop information

![Figure 5.28 Phases of the loop extraction process](image)

### 5.4 Chapter summary

In this chapter, a method was proposed for the extraction of stroke information from scanned images of handwritten signatures. We have overcome the problem of detecting sudden direction changes in the strokes by using an adaptive scanning circle radius and provided a locally optimal algorithm for estimating the drawing sequence of directly connected strokes. Experimental results show that – in terms of thinning – the accuracy of the Scanning Circle Algorithm is comparable to the best thinning algorithms examined. This makes the algorithm a suitable base for further use in a signature verification process.

We have proposed a heuristic algorithm for the solution of multidimensional assignment problem in the case of signature verification. A simplified version of
the algorithm ensures that features of sample signatures can also be assigned to the matched features of reference signatures.

We have described a way of characterizing loop and baseline information and provided algorithms for the extraction of these features from signature images. These algorithms, together with the classifier introduced in Section 3, cover the whole process of signature verification and serve as the basic building blocks of our signature verification system, to be discussed in the next chapter.
6. The application of the results

This thesis is the result of a seven-year-long research. During this period, a rich application system was built around the core of our signature verification system. This section briefly introduces some of the most important applications.

Related publications: [101] [102] [103] [104] [105] [106] [107] [108] [109].

6.1 Signature verification framework

Our signature verification framework currently consists of approximately 148,000 lines of code. About 10% of this codebase belongs to a single class library, which contains the most mature modules and the framework itself in a reusable and portable form.

The verification framework itself is an implementation of the model presented in Section 2.8. It coordinates the training and testing phases and provides benchmarking. Individual modules interact with the framework through the implementation of 4 well-defined interfaces belonging to the different processing phases. A detailed description of these interfaces can be found in Appendix B.

6.1.1 Architecture

One of the main goals of modern software design is to keep cohesion (within a component) high and coupling (between components) low [110]. As we have seen in section 2.6, state-of-the-art signature verifiers mostly lack these characteristics. Although they usually isolate some of the main modules, the core is often a highly coupled composition of different components, often incorporating loops and other complex constructs in the information flow. While this is not necessarily a design flaw, it makes the objective evaluation and the comparison of different systems hard and the simultaneous development of modules almost impossible.

Our model reduces coupling to a minimum. Only signature files and their corresponding metadata are passed between components. The input and output parameters of the interface functions are restricted to the minimum without limiting the functionality of the modules.

The data flow is well defined without any feedback among the processing phases. This structure allows independent development and testing of modules. Although dependencies are possible (e.g. a matching module may depend on a specified feature extraction module) the components implementing the same interfaces are interchangeable. In addition, several components (e.g. feature extraction algorithms) can be used simultaneously to create more sophisticated input for the classification module.

In the next subsections we shall present examples of how this model can be applied to aid the implementation and testing of various signature verification tasks.
6.1.2 A modular system

A generic signature verification framework has been established using the previously introduced architecture. Currently, the system supports the addition of:

- custom preprocessing steps,
- custom feature extraction functions,
- custom feature matching functions,
- custom classifiers.

The architecture allowed us to create benchmarks for each of the implemented components and thereby not only measure their effects on the global verification results, but to compare them to each other. Figure 6.1, Figure 6.2, and Figure 6.3 illustrate this functionality.

Currently, we have working modules for performing the following tasks:

Preprocessing

- cropping
- removing underlines (3 implementations) [111] [112] [113]
- noise removal
- skeletonization (2 implementations)
- scale transformation
- rotation normalization
- Matlab wrapper

Feature extraction

- extraction of strokes based on the skeleton
- extraction of endpoints based on the skeleton
- extraction of junctions based on the skeleton
- extraction of strokes based on the original image [114]
- extraction of intensity information from strokes
- extraction of line endings [115]
- extraction of slopes [115]
- extraction of baselines (2 implementations) [96]
- extraction of loops (2 implementations) [100]
- extraction of tremor information [115]
- extraction of basic size information
- extraction of lower and upper envelopes [112] [116]
- extraction of signature angle [115]
Matching
- matching of strokes [114]
- matching of baselines [96]
- matching of loops [100]
- generic matching algorithm (2 implementations) [117]

Classification
- threshold based classifier
- ellipsoid classifier [113]
- quartile based classifier
- random classifier
- neural network based classifier (3 implementations) [118] [119]
- statistics based classifier (2 implementations) [113]

6.1.3 Education
The above results could not have been reached without the modular nature of the system. It allowed students to get engaged in the task of signature verification without having to implement a whole verification system on their own. 15 Master’s thesis, 10 bachelor’s thesis and three Scientific Students’ Associations Conference papers are the fruit of their work. All of them were created under my supervision.

Figure 6.1 Module selection and bulk processing
The Signature Manager (SigMan) [120] application was created by Bence Kutasi to assist in the visual analysis of signatures and features. The highly modular Windows Presentation Foundation (WPF) based application is able to visualize the previously extracted features beside the signature. All stored
descriptive information about the signature and its features can be edited either directly in XML or in a more structured way through a property editor interface (Figure 6.4).

Figure 6.4 Screenshots from the SigMan application

6.3 Silverlight Frontend

To allow other researchers, students and visitors to experiment with our system, Mátýás Pálfalvi created a public interactive frontend [121], available on the homepage of the Department of Automation and Applied Informatics at BME (Figure 6.5).

The tool provides a simple set of original and forged signatures to demonstrate the capabilities of the system swiftly, but one can also upload (through dragging and dropping) his/her own signatures for testing.
6.4 WPF Frontend

The user interfaces used during development and testing are overcrowded and complex. The WPF based user interface created by Daniel Nagy [122] demonstrates that the complex inner logic of the system can be presented in a clear and user-friendly way (Figure 6.6).

6.5 Verification reports

One of the main aims of our research is not only to provide a commercial application for signature verification, but also to deliver meaningful details about the reasons behind a given decision.

The whole model of our system was designed to support this scenario (in contrast to many other similar systems), because:

- The feature set is based on the same features, as used in forensic document examination.
Each of the features is matched independently, and each of the matched feature groups is evaluated independently.

As a result of these design principles, each signature test conducted by our system also tells us which of the measured features deviated significantly from the usual normal distribution of the signer’s features.

As a unique feature among off-line signature verifiers, our system is able to create a report about the verification results of a signature to explain the decision (Figure 6.7). Instead of abstract results, the report provides a detailed visual feedback about the main processing results as well as statistical parameters for each feature. Using these results, a human expert can easily confirm or question the decision of the system. The current implementation of our reporting component was created by Zoltán Berceli [117] and Mátyás Pálfalvi [121].

Figure 6.7 Parts of the verification report
7. Conclusions

7.1 Summary

In this thesis, we have introduced the problems in feature-based off-line signature verification and proposed a framework that provides a quantitative way to characterize this problem class. By using normality tests, we verified that baseline and loop feature properties can efficiently be approximated with a normal distribution. We have shown that these assumptions can be used to define an acceptance threshold for a feature property that minimizes the average error rate (Lemma 1). We have also shown that when the count of reference signatures is low, the sample size should also be included in our calculations (Lemma 2). Using these results, we were able to predict the average error rate of our verification system (Lemma 3).

By our experiments, we have demonstrated that local features can successfully be used with our system to distinguish original signatures from forgeries with error rates similar to those of our estimations.

Baselines and loops were found to be good features for characterizing signatures written in Latin writing; however, they were insufficient in the case of Chinese writing. Currently, we are studying several other features that could extend the field of applicability and improve the accuracy of our algorithm. Furthermore, we are experimenting with different methods to reduce the possible dependencies among feature properties.

The results discussed in this work are summarized with three theses. I have proven these results with engineering and mathematical methods and have illustrated their practical relevance in engineering applications. Furthermore, I have published these results in several scientific forums.

**Thesis I**

I have shown that

- the distribution of several feature values can be approximated with normal distribution. The intuitive explanation was confirmed by experimental results. The statement has been also extended for forged feature values, which follow the same distribution but usually with greater variance.

- the average error rate of a classifier with more than 30 samples can be approximated with

\[
AER = \frac{\alpha + \beta}{2} = \frac{1}{2} + \Phi\left(\frac{w}{q}\right) - \Phi(w) \tag{7.1}
\]

- the previous error rate is always smaller than 0.5 when \( q > 1 \)

- in the previous model, the sum of the probabilities of false rejection and false acceptance is minimized when

\[
w = \pm \sqrt{2\ln q} \frac{q^2}{\sqrt{q^2 - 1}} \tag{7.2}
\]
CONCLUSIONS

- approximating the distributions with Student’s t-distributions, the error is minimized if

\[ w = \pm \sqrt{\frac{q^2 - q^2}{2n - q^2} \left( \frac{n^2 - 1}{n} \right)} \quad (7.3) \]

- If each of the feature properties is either rejected or accepted, and \( \alpha \) and \( \beta \) are known, the number of rejected features required to reject a signature while minimizing the average error rate is:

\[ l = k \ln \left( \frac{\ln(1-\alpha) - \ln(\beta)}{\ln(1-\alpha) - \ln(1-\beta) - \ln(\alpha)} \right) \quad (7.4) \]

- the lowest achievable average error rate of a signature verification system can be predicted based on the number of reference signatures used to train the system \((n)\), the number of independent feature properties \((k)\) and the quality of the forged features \((q)\) as follows:

\[
AER(q, n, k) = \frac{\Sigma_{j=0}^{l(i)} \binom{k}{j} \beta^{k-j} (1-\beta)^j + \Sigma_{j=1}^{k} \binom{k}{j} \alpha^{l(1-\alpha)^{k-j}}}{2}
\]

\[ w = \pm \sqrt{\frac{q^2 - q^2}{2n - q^2} \left( \frac{n^2 - 1}{n} \right)} \quad (7.6) \]

Related publications: [123] [124] [8] [101] [125] [126] [127] [128] [57] [129].

Thesis II
I have proved that

- given the constraints

\[ q > 1, k \geq 1, n > 2, n, k \in \mathbb{Z}, q \in \mathbb{R} \quad (7.7) \]

and provided that the number of reference samples \((n_0)\) and the forgery quality \((q_0)\) is given the function \(AER(q_0, n_0, k)\) is decreasing.

- given the constraints

\[ q > 1, k = 1, n > 2, n, k \in \mathbb{Z}, q \in \mathbb{R} \quad (7.8) \]

and provided that the number of observed features \((k_0)\) and the forgery quality \((q_0)\) is given the function \(AER(q_0, n, k_0)\) is strictly decreasing.

I have formulated Conjecture 4.1 and provided simulations to support it.

- Supposing that the value of the actual forgery quality \((q_{act})\) falls in the interval \((1; q_{max})\), there is an ideal value for estimated forgery quality \((q_{est})\) that minimizes the error resulting from the suboptimal choice of \(w\).

Related publications: [123] [124] [8] [101] [125] [126] [127] [128] [57] [129].
Thesis III

\begin{itemize}
  \item I have given a heuristic stroke extraction algorithm and provided experimental evidence to show that it is able to achieve comparable accuracy to the thinning algorithms with the best test results.
  \item I have proposed a heuristic algorithm for the solution of the multidimensional assignment problem for off-line signature verification. I have defined a simplified version of the algorithm for matching a sample signature to the reference signatures.
  \item I have described a way of characterizing baseline information and provided algorithms for the extraction of these features from signature images.
  \item I have described a way of characterizing loop information and provided algorithms for the extraction of these features from signature images.
\end{itemize}

Related publications: [101] [130] [102] [125] [131] [127] [104] [105] [106] [107] [132] [133] [134] [108] [74] [135] [27] [136] [137].

7.2 Future work

Future work can be divided into two main categories. First, there are several smaller questions that we are confident we can answer with a reasonable amount of effort.

\begin{itemize}
  \item Our current framework takes advantage of relatively few preprocessing steps. We have seen during our research that noise, or different parts of the signed form may be a source of error in further processing. We suspect that other transformations, like size or rotation normalization could also improve the accuracy of the matching algorithm. In the future we would like to investigate these claims more thoroughly.
  \item As feature extraction algorithms are mainly heuristic algorithms, they make mistakes often. On the other hand, by testing them on large and heterogeneous datasets, a large part of these errors can be systematically eliminated. The improvement of our existing feature extraction algorithms is currently one of our top priorities.
\end{itemize}

There are also two major questions that will probably require much more investigation.

\begin{itemize}
  \item One of the major presumptions of our current classification algorithm is the independence of features. As we have seen, as the number of features increases, their interdependency grows, which reduces the accuracy of our model for larger feature counts. To tackle this, either new feature types – mostly independent from the currently analyzed ones – should be introduced, or detected feature properties should be transformed to more independent values. We have already done some research in this field, but our current results are not satisfactory enough to draw a clear conclusion.
\end{itemize}
The correct reconstruction of strokes could be a second major step towards an accurate off-line signature verifier. As we have seen, state of the art thinning algorithms cannot cope with overlapping strokes. If this problem could be resolved, the matching problem could undergo a major simplification together with several feature extraction algorithms. There are several promising attempts to do this [45] [74], but they still have a long way to go.

In the literature, the most important (and usually only) parameter used to characterize a verification system is the error rate it can achieve. Although the improvement of this parameter is really the main goal of signature verification, it should be pointed out that this error rate is actually an aggregation of errors committed in previous processing steps. Hopefully, the achievements of this thesis will move the focus of researchers – at least to an infinitesimal extent – from the final results towards the more thorough analysis of individual processing steps and thereby help us better understand the mathematics of off-line signature verification.
Appendix A  Detailed proofs

A.1 Detailed proof of Proposition 3.2

Proposition: Assuming, that the distribution of feature properties can be approximated by normal distributions with known parameters, the sum of the probabilities of false rejection and false acceptance is minimized when

\[
w = \pm \sqrt{2} \sqrt{\ln q} \frac{q}{\sqrt{q^2 - 1}}
\]

Proof: The error is minimized if

\[
\left( \frac{1}{2} + \Phi \left( \frac{w}{\sqrt{q}} \right) - \Phi (w) \right)' = 0
\]

This is the case if

\[
f_o(x) = f_f(x)
\]

\[
\frac{1}{\sqrt{2\pi s_o^2}} e^{-\frac{1}{2} \left( \frac{x - m}{s_o} \right)^2} = \frac{1}{\sqrt{2\pi s_f^2}} e^{-\frac{1}{2} \left( \frac{x - m}{s_f} \right)^2}
\]

Using Equation 3.1 and Equation 3.4:

\[
\frac{1}{s} e^{-\frac{1}{2} \left( \frac{x - (m + ws)}{s} \right)^2} = \frac{1}{qs} e^{-\frac{1}{2} \left( \frac{x - (m + ws)}{qs} \right)^2}
\]

\[
e^{-\frac{1}{2} w^2} = \frac{1}{q} e^{-\frac{1}{2} \left( \frac{w}{q} \right)^2}
\]

\[
q = e^{\frac{1}{2} \left( \frac{w^2 - w^2}{q^2} \right)}
\]

\[
\ln q = \frac{w^2}{2} \left( \frac{q^2 - 1}{q^2} \right)
\]

\[
w = \pm \sqrt{\frac{2}{q^2 - 1}} \ln q = \pm \sqrt{2} \sqrt{\ln q} \frac{q}{\sqrt{q^2 - 1}}
\]
A.2 Detailed proof of Lemma 3.1

Lemma:

Let $X_1, X_2, ..., X_n$ be independent and identically normally distributed random variables having Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ with unknown mean and variance. If $X_{n+1}$ is a random variable independent of $X_1, X_2, ..., X_n$ and from the same distribution then its density function is

$$f(x) = \frac{1}{\hat{s}_n \sqrt{1 + \frac{1}{n}}} t_{n-1} \left( \frac{x - \hat{m}_n}{\hat{s}_n \sqrt{1 + \frac{1}{n}}} \right)$$

where $n > 1$, $\hat{m}_n$ denotes the sample mean and $\hat{s}^2_n$ denotes the sample variance of the first $n$ observations.

Proof:

It is known that if the elements of two sets of independent and identically distributed random variables $S_1$ and $S_2$ have the same Gaussian distribution, then

$$z = \frac{\bar{S}_1 - \bar{S}_2}{S_p \sqrt{\frac{1}{1} + \frac{1}{n}}} \sim t_{n-1}$$

follows Student t-distribution with $n - 1$ degrees of freedom, where $\bar{S}_1$ and $\bar{S}_2$ denote the sample means and $S_p$ is the pooled standard deviation, a weighted average of the standard deviations of the two samples. This is the base of the two sample t-test for equality of expectations for unequal sample count and equal but unknown variance.

Choosing $S_1 = \{x\}$ and $S_2 = \{X_1, X_2, ..., X_n\}$ we get:

$$\frac{\bar{S}_1 - \bar{S}_2}{S_p \sqrt{\frac{1}{1} + \frac{1}{n}}} = \frac{x - \hat{m}_n}{\hat{s}_n \sqrt{1 + \frac{1}{n}}} = z \sim t_{n-1}$$

Since we know the density of $z$ and $x$ is just a linear function of $z$ given $X_1, X_2, ..., X_n$, the density of $x$ can be calculated as simple transformation of measures. If $x = g(z)$ and $g: R \rightarrow R$ is invertible then

$$f_x(x) = f_z(g^{-1}(x)) |g^{-1}(x)'|$$

We can define $g(z)$ as:

$$g(z) = z * \hat{s}_n \sqrt{1 + \frac{1}{n} + \hat{m}_n}$$
from which:

\[ g^{-1}(x) = \frac{x - \hat{m}_n}{\hat{s}_n \sqrt{1 + \frac{1}{n}}} \]

As \( z \) has a distribution \( t_{n-1} \) the distribution of \( x \) can be calculated as:

\[ f_x(x) = f_z(g^{-1}(x))|g^{-1}(x)'| = \frac{1}{\hat{s}_n \sqrt{1 + \frac{1}{n}}} t_{n-1} \left( \frac{x - \hat{m}_n}{\hat{s}_n \sqrt{1 + \frac{1}{n}}} \right) \]

A.3 Detailed proof of Proposition 3.3

**Proposition:**

Assuming, that the distribution of feature properties can be approximated by normal distributions with unknown parameters the sum of the probabilities of false rejection and false acceptance is minimized when

\[ w = \pm \sqrt{\left( \frac{q^2 - q^2 + \frac{q^2}{n}}{q^2 - q^2 + \frac{q^2}{n}} \right) \left( \frac{n^2 - 1}{n} \right)} \]

**Proof:**

Based on Lemma 3.1, both the distributions of original and forged samples should be described based on Student’s t-distribution.

\[ t(x) = \frac{1}{\hat{s}_n \sqrt{1 + \frac{1}{n}}} t_{n-1} \left( \frac{x - \hat{m}_n}{\hat{s}_n \sqrt{1 + \frac{1}{n}}} \right) \]

Where \( t_{n-1} \) stands for the probability density function of the t-distribution with \( n - 1 \) degrees of freedom.

Let \( t_o(x) \) and \( t_f(x) \) denote the probability distribution function and \( T_o(x) \) and \( T_f(x) \) denote the cumulative distribution function of the original and forged samples, respectively.

In this case

\[ AER = \frac{\alpha + \beta}{2} = \frac{1}{2} + T_o(\hat{m}_n - w\hat{s}_n) - T_f(\hat{m}_n - w\frac{\hat{s}_n}{q}) \]
The error is minimized if

\[
\left( \frac{1}{2} + T_o(\bar{m}_n - w\hat{s}_n) - T_f(\bar{m}_n - w\hat{s}_n) \right) = 0
\]

\[
t_o(x) = t_f(x)
\]

\[
\frac{1}{\hat{s}_n \sqrt{1 + \frac{1}{n}}} \left( \frac{w\hat{s}_n}{\hat{s}_n \sqrt{1 + \frac{1}{n}}} \right) = \frac{1}{q\hat{s}_n \sqrt{1 + \frac{1}{n}}} \left( \frac{w\hat{s}_n}{\hat{s}_n \sqrt{1 + \frac{1}{n}}} \right)
\]

\[
t_{n-1} \left( \frac{w}{\sqrt{1 + \frac{1}{n}}} \right) = \frac{1}{q} t_{n-1} \left( \frac{w}{\sqrt{1 + \frac{1}{n}}} \right)
\]

\[
\frac{1}{\sqrt{n\pi}} \frac{\Gamma \left( \frac{n}{2} \right)}{\Gamma \left( \frac{n - 1}{2} \right)} \cdot \left( 1 + \frac{\frac{w^2}{2}}{1 + \frac{1}{n}} \right)^{-\frac{1}{2}(n)}
\]

\[
= \frac{1}{q} \frac{1}{\sqrt{n\pi}} \frac{\Gamma \left( \frac{n}{2} \right)}{\Gamma \left( \frac{n - 1}{2} \right)} \cdot \left( 1 + \frac{\frac{w^2}{2}}{1 + \frac{1}{n}} \right)^{-\frac{1}{2}(n)}
\]

\[
\left( 1 + \frac{\frac{w^2}{2}}{1 + \frac{1}{n}} \right)^{-\frac{n}{2}} = \frac{1}{q} \left( 1 + \frac{\frac{w^2}{2}}{1 + \frac{1}{n}} \right)^{-\frac{n}{2}}
\]

\[
\left( 1 + \frac{\frac{w^2}{2}}{1 + \frac{1}{n}} \right)^{-\frac{n}{2}} = \frac{1}{q} \left( 1 + \frac{\frac{w^2}{2}}{1 + \frac{1}{n}} \right)^{-\frac{n}{2}}
\]

\[
1 + \frac{nw^2}{n^2 - 1} = q^{\frac{n}{2}} \left( 1 + \frac{1}{q^{n} w^2} \right)
\]

\[
1 + \frac{nw^2}{n^2 - 1} = q^{\frac{n}{2}} + q^{\frac{n}{2}} \left( 1 + \frac{1}{q^{n} w^2} \right)
\]

\[
1 - q^{\frac{n}{2}} = q^{\frac{n}{2}} \left( 1 + \frac{1}{q^{n} w^2} \right)
\]

\[
w = \pm \sqrt{\frac{1 - \frac{q^{n}}{2}}{q^{\frac{n}{2}} \frac{1}{n} \frac{n}{n^2 - 1} - \frac{1}{n^2 - 1}}} = \pm \sqrt{\left( \frac{q^2 - q^{\frac{n}{2}}}{q^{\frac{n}{2}} - q^2} \right) \left( \frac{n^2 - 1}{n} \right)}
\]
A.4 The calculation of $\alpha$ and $\beta$ based on Proposition 3.3

\[
t_o(x) = \frac{1}{s_n \sqrt{1 + \frac{1}{n}}} \frac{1}{\sqrt{\pi(n-1)}} \frac{\Gamma(n)}{\Gamma\left(\frac{n}{2}\right)} \left( 1 + \frac{\left(\frac{x - \hat{m}_n}{s_n \sqrt{1 + \frac{1}{n}}}\right)^2}{\frac{n}{2}} \right)
\]

\[
t_f(x) = \frac{1}{q_s^2 \sqrt{1 + \frac{1}{n}}} \frac{1}{\sqrt{\pi(n-1)}} \frac{\Gamma(n)}{\Gamma\left(\frac{n}{2}\right)} \left( 1 + \frac{\left(\frac{x - \hat{m}_n}{q_s \sqrt{1 + \frac{1}{n}}}\right)^2}{\frac{n}{2}} \right)
\]

\[
\alpha = P(m - ws > X) + P(m + ws < X) = 2T_o(m_n - w_s_n)
\]

\[
= 2 \int_{-\infty}^{\infty} t_o(x)dx = \frac{2}{s_n \sqrt{1 + \frac{1}{n}}} \frac{1}{\sqrt{\pi(n-1)}} \frac{\Gamma(n)}{\Gamma\left(\frac{n}{2}\right)}
\]

\[
\int_{-\infty}^{\infty} \left( 1 + \frac{\left(\frac{x - \hat{m}_n}{s_n \sqrt{1 + \frac{1}{n}}}\right)^2}{\frac{n}{2}} \right) dx
\]

\[
\beta = P(m - ws < X < m + w_s_o) = 1 - 2T_f(m_n - w_s_n)
\]

\[
= 1 - 2 \int_{-\infty}^{\infty} t_f(x)dx = 1 - 2 \frac{1}{q_s \sqrt{1 + \frac{1}{n}}} \frac{1}{\sqrt{\pi(n-1)}} \frac{\Gamma(n)}{\Gamma\left(\frac{n}{2}\right)}
\]
\[
\int_{-\infty}^{\bar{m}_n - w \hat{s}_n} \left( 1 + \frac{\left( \frac{x - \bar{m}_n}{q \hat{s}_n \sqrt{1 + \frac{1}{n}}} \right)^2}{n - 1} \right)^{-\frac{n}{2}} \, dx
\]
A.5 Detailed proof of Proposition 3.4

**Proposition:** If each of the feature properties is either rejected or accepted, and \( \alpha \) and \( \beta \) are known, the number of rejected features required to reject a signature while minimizing the average error rate is:

\[
l = k \frac{\ln(1-\alpha) - \ln(\beta)}{\ln(1-\alpha) - \ln(\beta) + \ln(1-\beta) - \ln(\alpha)}
\]

**Proof:** We can find the point where

\[
P(\theta_f | I) = P(\theta_o | I)
\]

\[
\frac{(1 - \beta)^l \beta^{k-l}}{(1 - \beta)^l \beta^{k-l} + \alpha^l (1 - \alpha)^{k-l}} = \frac{\alpha^l (1 - \alpha)^{k-l}}{(1 - \beta)^l \beta^{k-l} + \alpha^l (1 - \alpha)^{k-l}}
\]

\[
(1 - \beta)^l \beta^{k-l} = \alpha^l (1 - \alpha)^{k-l}
\]

\[
\frac{(1 - \beta)^l}{\alpha^l} = \frac{(1 - \alpha)^{k-l}}{\beta^{k-l}}
\]

\[
\frac{(1 - \beta)^l}{\alpha^l} = \frac{(1 - \alpha)^k}{\beta^k}
\]

\[
\frac{(1 - \beta)^l (1 - \alpha)^l}{\alpha^l \beta^l} = \frac{(1 - \alpha)^k}{\beta^k}
\]

\[
l = k \log_{\frac{(1-\alpha)(1-\beta)}{\alpha\beta}} \frac{1 - \alpha}{\beta} = k \frac{\ln \frac{1 - \alpha}{\beta}}{\ln \frac{(1 - \alpha)(1 - \beta)}{\alpha\beta}}
\]

\[
= k \frac{\ln(1 - \alpha) - \ln(\beta)}{\ln(1 - \alpha) - \ln(\beta) + \ln(1 - \beta) - \ln(\alpha)}
\]
A.6 Details for the proof of Proposition 4.1

A.6.1 Details of equation 4.16:

\[
AER(k) = \frac{\sum_{j=0}^{l} \binom{k}{j} \beta^{k-j}(1 - \beta)^j + \sum_{i=|l|+1}^{k} \binom{k}{i} \alpha^i (1 - \alpha)^{k-i}}{2}
\]

\[
AER(k + 1) = \frac{\sum_{j=0}^{l} \binom{k+1}{j} \beta^{k+1-j}(1 - \beta)^j + \sum_{i=|l|+1}^{k+1} \binom{k+1}{i} \alpha^i (1 - \alpha)^{k+1-i}}{2}
\]

Step 1:

\[
AER(k) > AER(k + 1)
\]

Step 2:

\[
\sum_{j=0}^{l} \binom{k}{j} \beta^{k-j}(1 - \beta)^j + \sum_{i=|l|+1}^{k} \binom{k}{i} \alpha^i (1 - \alpha)^{k-i}
\]

\[
\sum_{j=0}^{l} \binom{k+1}{j} \beta^{k+1-j}(1 - \beta)^j + \sum_{i=|l|+1}^{k+1} \binom{k+1}{i} \alpha^i (1 - \alpha)^{k+1-i}
\]

Step 3:

\[
\sum_{j=0}^{l} \binom{k}{j} \beta^{k-j}(1 - \beta)^j - \sum_{j=0}^{l} \binom{k+1}{j} \beta^{k+1-j}(1 - \beta)^j + \\
> \sum_{i=|l|+1}^{k+1} \binom{k+1}{i} \alpha^i (1 - \alpha)^{k+1-i} - \sum_{i=|l|+1}^{k} \binom{k}{i} \alpha^i (1 - \alpha)^{k-i}
\]
First, we calculate the sum for $\beta$:

$$
\sum_{j=0}^{l} \binom{k}{j} \beta^{k-j}(1 - \beta)^{j} - \sum_{j=0}^{l} \binom{k+1}{j} \beta^{k+1-j}(1 - \beta)^{j} =
$$

$$
\binom{k}{0} \beta^{k} - \binom{k}{1} \beta^{k-1}(1 - \beta) + \binom{k}{2} \beta^{k-2}(1 - \beta)^{2} - \cdots
$$

Coefficients for $\beta^{k+1}$:

$$
- \binom{k+1}{0} \beta^{k+1} - \binom{k+1}{1} \beta^{k+1-1}(1 - \beta) - \binom{k+1}{2} \beta^{k+1-2}(1 - \beta)^{2} - \cdots
$$

Coefficients for $\beta^{k}$:

$$
\binom{k}{0} - \binom{k}{1} + \binom{k}{2} - \binom{k}{3} + \cdots - (-1)^{l} \binom{k}{l} =
$$

Coefficients for $\beta^{k-1}$: $- (-1)^{l} \binom{k}{l} \binom{l+1}{2}$

Coefficients for $\beta^{k-2}$: $- (-1)^{l} \binom{k}{l} \binom{l+1}{3}$

Coefficients for $\beta^{k-3}$: $- (-1)^{l} \binom{k}{l} \binom{l+1}{4}$

... 

From the above coefficients:

$$
\sum_{j=0}^{l} \binom{k}{j} \beta^{k-j}(1 - \beta)^{j} - \sum_{j=0}^{l} \binom{k+1}{j} \beta^{k+1-j}(1 - \beta)^{j} = \binom{k}{l} \beta^{k-l}(1 - \beta)^{l}
$$
Similarly:

Coefficients for $\alpha^{k+1}$: $(-1)^{l} \binom{k}{l}$

Coefficients for $\alpha^{k}$: $-(-1)^{l} \binom{k}{l} \binom{k-l}{l}$

Coefficients for $\alpha^{k-1}$: $(-1)^{l} \binom{k}{l} \binom{k-l}{l}$

Coefficients for $\alpha^{k-2}$: $-(-1)^{l} \binom{k}{l} \binom{k-l}{l}$

\[ \ldots \]

From the above coefficients:

\[
\sum_{i=l+1}^{k+1} \binom{k+1}{i} \alpha^{i} (1 - \alpha)^{k+1-i} - \sum_{i=l+1}^{k} \binom{k}{i} \alpha^{i} (1 - \alpha)^{k-i} = \binom{k}{l} \alpha^{l+1} (1 - \alpha)^{k-l}
\]

Therefore the original inequation can be written in this form:

\[
\binom{k}{l} \beta^{k-l} (1 - \beta)^{l+1} > \binom{k}{l} \alpha^{l+1} (1 - \alpha)^{k-l}
\]

\[
(1 - \beta) (\beta^{k-l} (1 - \beta)^{l}) > \alpha (\alpha^{l} (1 - \alpha)^{k-l})
\]

\[
\frac{1 - \beta}{\alpha} > \frac{(\alpha^{l} (1 - \alpha)^{k-l})}{(\beta^{k-l} (1 - \beta)^{l})}
\]
A.6.2 Details for equation 4.23:

Let

\[ u = \log_{\frac{(1-\beta)(1-\alpha)}{\alpha\beta}} \frac{1-\alpha}{\beta} \]

\[ \left( \frac{1-\beta}{\alpha} \right)^{l+u} \left( \frac{1-\alpha}{\beta} \right)^{l+u} < \left( \frac{1-\beta}{\alpha} \right)^{l} \left( \frac{1-\alpha}{\beta} \right)^{l} \]

\[ \left( \frac{1-\beta}{\alpha} \right)^{l} \left( \frac{1-\alpha}{\beta} \right)^{l} \left( \frac{1-\beta}{\alpha} \right)^{u} \left( \frac{1-\alpha}{\beta} \right)^{u} < \left( \frac{1-\beta}{\alpha} \right)^{l} \left( \frac{1-\alpha}{\beta} \right)^{l} \]

\[ \left( \frac{1-\beta}{\alpha} \right)^{l} \left( \frac{1-\alpha}{\beta} \right)^{l} \left( \frac{(1-\beta)(1-\alpha)}{\alpha\beta} \right)^{l} < \left( \frac{1-\beta}{\alpha} \right)^{l} \left( \frac{1-\alpha}{\beta} \right)^{l} \]

\[ \left( \frac{1-\beta}{\alpha} \right)^{l} \left( \frac{1-\alpha}{\beta} \right)^{l} \left( \frac{1-\alpha}{\beta} \right)^{l} < \left( \frac{1-\beta}{\alpha} \right)^{l} \left( \frac{1-\alpha}{\beta} \right)^{l} \]

A.6.3 Details of equation 4.27:

\[ \sum_{j=0}^{l} \binom{k}{j} \beta^{k-j}(1-\beta)^{j} - \sum_{j=0}^{l+1} \binom{k+1}{j} \beta^{k+1-j}(1-\beta)^{j} = -\binom{k}{l+1} \beta^{k-l}(1-\beta)^{l+1} \]

\[ \sum_{i=l+2}^{k+1} \binom{k+1}{i} \alpha^{i}(1-\alpha)^{k+1-i} - \sum_{i=l+1}^{k} \binom{k}{i} \alpha^{i}(1-\alpha)^{k-i} = -\binom{k}{l+1} \alpha^{l+1}(1-\alpha)^{k-l} \]

\[ -\binom{k}{l+1} \beta^{k-l}(1-\beta)^{l+1} \geq -\binom{k}{l+1} \alpha^{l+1}(1-\alpha)^{k-l} \]

\[ \beta^{k-l}(1-\beta)^{l+1} \leq \alpha^{l+1}(1-\alpha)^{k-l} \]

\[ \frac{1-\beta}{\alpha} \leq \frac{(\alpha^{l}(1-\alpha)^{k-l})}{(\beta^{k-l}(1-\beta)^{l})} \]
A.7 Scripts for Proposition 4.2

This section provides the scripts used in Wolfram Mathematica [138] to model the equations of the proposition.

The cumulative distribution function for Student’s t-distribution, with scale $q$ and location 0 can be described as follows:

$$T[x_, n_, q_] = \frac{1}{2} + \frac{x}{q \sqrt{1+1/n}} \Gamma[(n)/2] \text{Hypergeometric2F1}[1/2, (n)/2, 3/2, -(x/(q \sqrt{1 + 1/n}))^2/(n - 1)]/(\sqrt{\pi (n - 1)} \Gamma[(n - 1)/2])$$

Decision threshold (Equation 3.24):

$$w[n_, q_] = \sqrt{(q^2 - q^n(2 + 2/n))/(q^n(2/n) - q^2)(n^2 - 1)/n}$$

$AER$ for a single feature (Equation 3.21):

$$F[n_, q_] = (2 T[-w[n, q], n, 1] + 1 - 2 T[-w[n, q], n, q])/2$$

Partial derivative of $AER$ ($\frac{dAER(q, n, 1)}{dn}$):

$$f[n_, q_] = D[F[n, q], n]$$

Output

Plot3D[$F[n, q]$, {n, 2, 90}, {q, 1.001, 10}]

Plot3D[$f[n, q]$, {n, 2, 90}, {q, 1.001, 10}]
Appendix B  The signature verification API

This is a compact introduction to the main classes of the signature verification framework. You will need the following three classes to load signature metadata from a predefined database structure and to perform different operations on them:

- **Signer, Signature**: These are data classes that represent all metadata currently available about a signer or a signature. In most of the cases, you should not instantiate these classes directly, but use the methods of IStorage to load them.
- **FileSystemStorage**: Provides access to signatures stored in a predefined folder structure in the file system. This is currently the only available implementation of IStorage.

The framework is designed to allow a loose coupling of several independent processing modules. Based on the type of the module you want to implement, you have to choose between the following interfaces:

- **ITransformation**: you should implement this interface if you want to make non-reversible changes on the signature image. The interface exposes a single method that should get the signature to process as a parameter. Any changes to the metadata should be directly performed on this object. To preserve changes made on the image itself, signature.SaveImage() must be called before leaving the method.

  ```csharp
  void Transform(Signature signature);
  ```

- **IFeatureExtraction**: you should implement this interface if you want to add information to signature metadata. The interface exposes a single method that should get the signature to process as a parameter. Changes are preserved in the signature object, the framework should take care of saving the information if required.

  ```csharp
  void Transform(Signature signature);
  ```

- **IModelFeatureExtraction**: you should implement this interface if you want to add information to signer metadata. The interface exposes two methods. The model parameter contains all signer specific information, while the next parameter contains information about available reference signatures for the signer. The third parameter may contain an array of sample signatures. This is also the main interface for implementing matching algorithms.

  ```csharp
  void Process(SignatureModel model, Signature[] signatures);
  void Process(SignatureModel model, Signature[] originalSignatures,
               params Signature[] sampleSignatures);
  ```

- **IClassification**: the primary interface of all classification modules. The classification should be implemented in two phases. First, the Train method is called with the matched feature values of the reference signatures. Then the Test method can be called multiple times with a feature vector obtained from the sample signature. The headers
parameter may contain the names of the feature values allowing the creation of more detailed ClassificationResult structure.

```csharp
FeatureHeader[] Train(double[,] featureValues);
FeatureHeader[] Train(double[,] values, FeatureHeader[] headers);
ClassificationResult Test(double[] featureValues);
```

The following sample demonstrates how the implementation of a simple classifier would look

```csharp
public double Check(string[] originalSignatureImageFiles, string sampleSignatureImageFile)
{
    // Create data objects from images
    Signature[] originals = originalSignatureImageFiles
        .Select(img => Signature.LoadFromFile(img)).ToArray();
    Signature sample = Signature.LoadFromFile(sampleSignatureImageFile);
    var all = originals.ToList();
    all.Add(sample);

    // Extract basic metadata, baseline and loop information
    foreach (Signature sig in all)
    {
        new BasicMetadataExtraction().Process(sig);
        new BaselineExtraction().Process(sig);
        new LoopExtraction().Process(sig);
    }

    // Match baselines and loops
    var model = new SignatureModel();
    new BaselineMatching().Process(model, originals.ToArray(), sample);
    new LoopMatching().Process(model, originals.ToArray(), sample);

    // Load the matched features for classification
    var loader = new FeatureLoader();
    double[,] features;
    FeatureHeader[] headers;
    loader.LoadFeatures(all.ToArray(), model, out features, out headers);

    // Classifier training with the reference signatures
    var classifier = new StatisticalClassifier();
    classifier.Train(
        features.GetPart(0,0,originalSignatureImageFiles.Length, 
                        features.GetLength(1)), headers);
    // Testing of sample signature
    var classificationResult = classifier.Test(
        features.GetRow(originalSignatureImageFiles.Length).ToArray());

    return classificationResult.MatchProbability;
}
Appendix C  SVC 2004 database samples

Jones  Mary  wishes

Joey  XY chae  Ray

Rice  Happy  David

Soan  Frank  Steve

Benny  Jan  James

Verna  debbie  Pete
8. Bibliography


[15] D. Muramatsu and T. Matsumoto, "Effectiveness of Pen Pressure,


[58] K. Huang and H. Yan, "Off-line signature verification using structural


Researchers on Computational Intelligence and Informatics (CINTI), 2008.


