

ISLAMIC UNIVERSITY OF TECHNOLOGY (IUT)

# OFFLINE SIGNATURE VERIFICATION USING FACTORIZED GRAPH MATCHING

AUTHOR

**A.B.M. Ashikur Rahman**

STUDENT ID: 144610

SUPERVISOR

**Md. Hasanul Kabir, PhD**

PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ISLAMIC UNIVERSITY OF TECHNOLOGY

**A thesis submitted to the Department of Computer Science and  
Engineering (CSE) in partial fulfilment of the requirements for the  
degree of M.Sc. Engineering in CSE**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING(CSE)

ISLAMIC UNIVERSITY OF TECHNOLOGY (IUT)

A SUBSIDIARY ORGAN OF THE ORGANIZATION OF ISLAMIC COOPERATION (OIC)

GAZIPUR 1704, DHAKA, BANGLADESH

NOVEMBER 2018





## RECOMMENDATION OF THE BOARD OF EXAMINERS

The work titled “**Offline Signature Verification using Factorized Graph Matching**”, submitted by **A.B.M. Ashikur Rahman (Student Id. 144610)** in the academic year 2017-2018 has been found satisfactory and accepted as partial fulfilment of the requirement for the degree of Master of Science in Computer Science and Engineering (M.Sc. Engg.(CSE)) on Monday, 26 November 2018.

1. \_\_\_\_\_  
**Md. Hasanul Kabir, PhD**  
Professor,  
Department of Computer Science and Engineering (CSE),  
Islamic University of Technology (IUT),  
Board Bazar, Gazipur-1704, Dhaka, Bangladesh.

**Chairman**  
(Supervisor)

2. \_\_\_\_\_  
**Muhammad Mahbub Alam, PhD**  
Professor & Head,  
Department of Computer Science and Engineering (CSE),  
Islamic University of Technology (IUT),  
Board Bazar, Gazipur-1704, Dhaka, Bangladesh.

**Member**  
(Ex-officio)

3. *Md. Kamrul Hasan*  
*26.11.2018*  
\_\_\_\_\_  
**Md. Kamrul Hasan, PhD**  
Professor,  
Department of Computer Science and Engineering (CSE),  
Islamic University of Technology (IUT),  
Board Bazar, Gazipur-1704, Dhaka, Bangladesh..

**Member**

4. *Rafiqul Islam*  
\_\_\_\_\_  
**Rafiqul Islam, PhD**  
Associate Professor,  
Department of Computer Science and Engineering (CSE),  
Dhaka University of Engineering and Technology (DUET),  
Gazipur-1700, Bangladesh.

**Member**  
(External)

# Declaration

I hereby declare that this dissertation entitled **Offline Signature Verification using Factorized Graph Matching** was carried out by me for the degree of **Master of Science in Computer Science and Engineering**, M.Sc. (Engg.) in CSE, under the guidance and supervision of Prof. Dr. Md. Hasanul Kabir, Islamic University of Technology, Gazipur, Dhaka, Bangladesh.

The findings put forth in this work are based on my research and understanding of the original works and they are not published anywhere in the form of books, monographs or articles. The other books, articles and websites, which I have made use of are acknowledged at the respective place in this thesis.

For the present thesis, which I am submitting to the University, no degree or diploma or distinction has been conferred on me before, either in this or in any other University.

## Author

---

A.B.M. Ashikur Rahman  
Student ID: 144610

Date: \_\_\_\_\_



# Abstract

Signature verification is used in verifying the claimed identity of a person through his/her chosen and previously registered signature. The signature's widespread acceptance by the public and niche applications (validating paper documents and use in banking applications) makes it a desirable biometric.

Signature is considered to be a behavioral biometric that encodes the ballistic movements of the signer and as such is difficult to imitate. On the other hand, compared to physical traits such as fingerprint, iris or face, a signature typically shows higher intra-class and time variability. Furthermore, as with passwords, a user may choose a simple signature that is easy to forge.

In this work, we present a state-of-the-art offline signature verification system that uses a fusion of complementary features, classifiers and preprocessing techniques, with the aim to explore the limits in signature verification accuracy.

Here we propose an efficient way to verify the identity of a claimed person. This proposed method uses graph matching for measuring the similarity between two sample signatures. For making it eligible to apply graph matching, features are computed by creating graph representation of the signature image. We propose to use direction information for edge feature to make the method more robust to rotation of the image. This direction is normalized to cope with the intra-class variability. For matching Factorized graph matching is used as it provides faster computation of point-wise correspondence. Based on the affinity score, a statistical approach is made for decision making. For validation Writer-Dependent approach is applied.

Extensive experiments have been conducted on a benchmark and publicly available Offline Signature Dataset (ICDAR 2011), where the proposed algorithm achieves approximately 12% average False Acceptance Rate (FAR) for signature verification. Furthermore, comparison with other promising works on the aforementioned databases demonstrates the enhanced performance and efficiency of the proposed method.

## Acknowledgements

It is an auspicious moment for me to submit my Master's thesis work by which I am eventually going to end my Master's study. At the beginning, I want to express our heart-felt gratitude to Almighty Allah for his blessings to bestow upon me which made it possible to complete this thesis research successfully. Without the mercy of Allah, I wouldn't be where I am right now. All thanks and praises be to Allah.

Secondly, I would like to thank my thesis supervisor, Dr. Md. Hasanul Kabir, Professor, Department of CSE, IUT, for his support and guidance on this thesis. Dr. Kabir has been an instrumental in this work and my career. He taught me how to do research, think critically, be a graduate student, and teach effectively. His all-time guidance, encouragement and continuous observation made the whole matter as a successful one. Without his continuous support, this thesis would not see the path of a proper itinerary of the research world.

It was my pleasure to get the cooperation and coordination from the Head of the Department, Professor Dr. Muhammad Mahbub Alam during various phases of the work. I am grateful to him for his constant and energetic guidance, constructive criticism and valuable advice. The faculty members of the CSE department of IUT helped make my working environment a pleasant one, by providing a helpful set of eyes and ears when problems arose.

I must thank the authors of Factorized Graph Matching, Feng Zhou and Fernando De la Torre for making their code publicly available. I also wish to take an opportunity to articulate my sincerest gratitude and heartiest thanks to Dr. Md. Kamrul Hasan, Professor, Department of CSE, IUT and Dr. Abu Raihan Mostofa Kamal, Professor, Department of CSE, IUT for their continuous support and encouragement to improve this thesis work.

I would like to thank the jury members of my thesis committee for the many interesting comments and criticism that helped improve this manuscript. Last but not the least, I am deeply grateful to my friends and family members for their love and unconditional support. This work would have never been completed without the consistent support and encouragement from them throughout my Master's program.

I am also thankful to the people whom I did not mention, but have a valuable contribution to this research.

Dedication  
**My respected Parents**  
*"Without whom none of my success would be possible"*

# Contents

Certificate	ii
Declaration	iii
Abstract	iv
Acknowledgements	v
List of Tables	ix
List of Figures	x
<b>1 Introduction</b>	<b>1</b>
1.1 Background	1
1.2 Problem Statement	2
1.3 Research Challenges	3
1.4 Thesis Objectives	4
1.5 Thesis Contributions	5
1.6 Thesis Organization	5
<b>2 Literature Review</b>	<b>6</b>
2.1 Feature Extraction	6
2.1.1 Geometric Features	7
2.1.2 Mathematical transformations	8
2.1.3 Texture features	8
2.1.4 Interest point matching	9
2.1.5 Graph Matching	10
2.2 Training Model	12
2.2.1 Hidden Markov Models	13
2.2.2 Support Vector Machines	13
2.2.3 Neural Networks and Deep Learning	13
<b>3 Proposed Method</b>	<b>15</b>
3.1 Pre-processing	15
3.1.1 Size normalization and centering	16
3.1.2 Gray-scale Conversion	16
3.1.3 Noise Removal	16
3.2 Keypoint Detection	16
3.3 Graph Formulation	17
3.4 Feature Extraction	19
3.5 Affinity Matrix Computation	21



3.6	Factorized Graph Matching (FGM)	22
3.7	Signature Data Storage	25
3.8	Decision Making	26
<b>4</b>	<b>Experimental Analysis</b>	<b>27</b>
4.1	Data Set and Experimental Setup	27
4.2	Performance Criterion	29
4.3	Performance Method Evaluation	30
4.3.1	Comparative Analysis	31
4.3.2	Comparison Regarding Computation Time	33
<b>5</b>	<b>Conclusion</b>	<b>35</b>
5.1	Summary of the Contributions	35
5.2	Future Works	35
	<b>Bibliography</b>	<b>37</b>
	<b>Publication List</b>	<b>42</b>

# List of Tables

2.1	Speed comparison of feature detectors on signature image . . . . .	12
4.1	Summary of DUTCH and CHINESE dataset . . . . .	29
4.2	List of methods used for comparative analysis . . . . .	32
4.3	Performance Comparison for Chinese Dataset . . . . .	32
4.4	Performance Comparison for Dutch Dataset . . . . .	32
4.5	Comparison based on Computation Time . . . . .	33

# List of Figures

1.1	Different types of forgery	2
1.2	Intra-class variability of Signature image	3
2.1	Block diagram of existing training methods	6
2.2	Block diagram of existing testing methods	6
2.3	Slope of line obtained from curve fitting of centre of gravity of each column.	7
2.4	Computation of fractal dimension	8
2.5	Log polar grids used in [1] varying the origin (a) Origin at the center (b) Origin at the top left corner	9
2.6	Block diagram of the proposed signature verification system. SDoGKeyGen: SDoG Keypoints Generation; SIFTDescGen: SIFT Descriptors Generation; TrasfComp: Transformation Computation and hypothesis rejection tests; BayesK: Bayes Classifier. [2]	9
2.7	Heat maps of some example specimen (genuine) signatures from two different authors (one genuine author in each row) showing the most stable (green) and the most unstable (red) parts along with the moderately stable parts (colors varying from green to red through blue) [3]	10
2.8	Signature preprocessing: a) original image, b) image after pepper noise removal, skew elimination, smoothing, and thinning, and c) final 64x128 normalized image.	11
2.9	Feature computation of each edge in the Adjacency matrix [4]	12
3.1	Steps of the proposed system	15
3.2	Effect of noise removal from signature Image	16
3.3	(a) A processed interest point and 16 pixels surrounding on it, (b) the demonstration of storing 16 values surrounding pixels in a vector form.	17
3.4	Output of Keypoint detection on Chinese Dataset	18
3.5	Output of Keypoint detection on Chinese Dataset	18
3.6	Delaunay Triangulation does not favor skinny triangles	19
3.7	Graph Formulation from keypoint, (a) Original image (b) Formulated graph	19
3.8	Wavelet response of keypoint neighborhood	20

3.9	Example of graph matching and related matrices. (a) Two synthetic graphs. (b) The 1 <sup>st</sup> graph's incidence matrix G1. (c) The 2 <sup>nd</sup> graph's incidence matrix G2. (d) The node affinity matrix Kp. (e) The edge distance affinity matrix Kq. (e) The edge direction affinity matrix Kq.	
	(g) The global affinity matrix K	22
3.10	Point wise correspondence in Chinese Dataset	25
3.11	Point wise correspondence in Dutch Dataset	25
4.1	Sample signature Images from Dutch Dataset; (a) Genuine Signature, (b) Forged sample	28
4.2	Sample signature Images from Chinese Dataset; (a) Genuine Signature, (b) Forged sample	29
4.3	Confusion Matrix for signature verification	30
4.4	: Comparison of FAR between two proposed methods	31
4.5	: Comparison of Accuracy between two proposed methods	31
4.6	Comparison of Accuracy and F-measure	33
4.7	Comparison of False Acceptance Rate (FAR) and False Rejection Rate (FRR)	33

# Chapter 1

## Introduction

In this chapter, we first present an overview of our thesis that includes the significance of the problem and the problem statement in detail. Then, we discuss different research challenges what we are going to face in the whole scenario. After that, we present our thesis objectives and contributions. The chapter ends with a short description of the organization of this thesis.

### 1.1 Background

Biometrics technology is used in a wide variety of security applications. The aim of such systems is to recognize a person based on physiological or behavioral traits. In the first case, the recognition is based on measurements of biological traits, such as the fingerprint, face, iris, etc. The later case is concerned with behavioral traits such as voice and the handwritten signature.

Biometric systems are mainly employed in two scenarios: verification and identification. In the first case, a user of the system claims an identity, and provides the biometric sample. The role of the verification system is to check if the user is indeed who he or she claims to be. In the identification case, a user provides a biometric sample, and the objective is to identify it among all users enrolled in the system. The handwritten signature is a particularly important type of biometric trait, mainly due to its ubiquitous use to verify a person's identity in legal, financial and administrative areas. One of the reasons for its widespread use is that the process to collect handwritten signatures is non-invasive, and people are familiar with the use of signatures in their daily life. Signature verification systems aim to automatically discriminate if the biometric sample is indeed of a claimed individual. In other words, they are used to classify query signatures as genuine or forgeries.

The forgeries in handwritten signatures have been categorized based on their characteristic features [5]. The three major types of forgeries are:

- **Random Forgery** The forger has no information about the user or his signature and uses his own signature instead. In this case, the forgery contains a different semantic meaning than the genuine signatures from the user, presenting a very different overall shape.

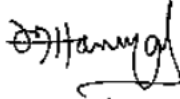
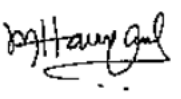
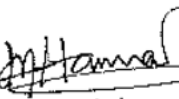
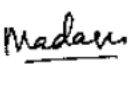
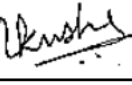
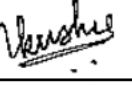
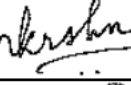

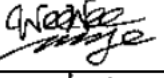
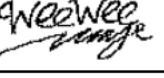
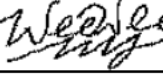
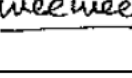
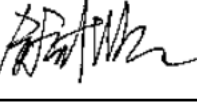
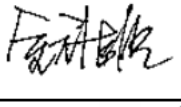
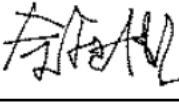
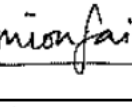
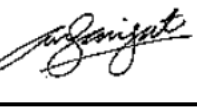
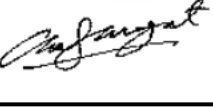
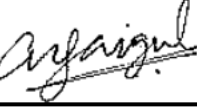
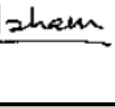
Genuine	Skilled forgery	Unskilled forgery	Random forgery
			
			
			
			
			

Figure 1.1: Different types of forgery

- Simple Forgery The signer imitates the signature in his own style without any knowledge of the spelling and does not have any prior experience. The imitation is preceded by observing the signature closely for a while.
- Skilled Forgery Undoubtedly, the most difficult of all forgeries is created by professional impostors or persons who have experience in copying the signature. For achieving this one could either trace or imitate the signature by hard way.

Depending on the acquisition method, signature verification systems are divided in two categories: online (dynamic) and offline (static). In the online case, an acquisition device, such as a digitizing table, is used to acquire the user's signature. The data is collected as a sequence over time, containing the position of the pen, and in some cases including additional information such as the pen inclination, pressure, etc. In offline signature verification, the signature is acquired after the writing process is completed. In this case, the signature is represented as a digital image.

## 1.2 Problem Statement

The problem of automatic handwritten signature verification is commonly modeled as a verification task: given a learning set  $L$ , that contains genuine signatures from a set of users, a model is trained. This model is then used for verification: a user claims an identity and provides a query signature  $X_{new}$ . The model is used to classify the signature as genuine (belonging to the claimed individual) or forgery (created by someone else). To evaluate the performance of the system, we consider a test set  $T$ , consisting of genuine signatures and forgeries. The signatures are acquired in an enrollment phase, while the second phase is referred to operations (or classification) phase. If a single model is used to classify images from any user, we refer to it as a





Figure 1.2: Intra-class variability of Signature image

writer-independent (WI) system. If one model is trained for each user, it is referred as a writer-dependent (WD) system. For WI systems, the common practice is to train and test the system with a different subset of users. In this case, we consider a development set  $D$  (which is used to train the WI model), and an exploitation set  $E$ , which represent the users enrolled to the system (and is further divided in  $L$  and  $T$ , as indicated above).

### 1.3 Research Challenges

Every domain comes with its own difficulties. People are used to verify signature images with bare eyes. Taking this task digitally and making it a automation provides a range of challenges. In order to get a usable verifying system, it is necessity to overcome those challenges. Researchers face a number of challenges in this domain. Some of them are-

- **High Intra-class variability**

One of the main challenges for the signature verification task is having a high intra-class variability. Compared to physical biometric traits, such as fingerprint or iris, handwritten signatures from the same user often show a large variability between samples. This problem is illustrated in Figure [1.2](#). This issue is aggravated with the presence of low inter-class variability when we consider skilled forgeries. These forgeries are made targeting a particular individual, where a person often practices imitating the user's signature. For this reason, skilled forgeries tend to resemble genuine signatures to a great extent.

- **Presence of partial knowledge**

Another important challenge for training an automated signature verification system is the presence of partial knowledge during training. In a realistic scenario, during training we only have access to genuine signatures for the users enrolled to the system. During operations, however, we want the system not only to be able to accept genuine signatures, but also to reject forgeries. This

is a challenging task, since during training a classifier has no information to learn what exactly distinguishes a genuine signature and a forgery for the users enrolled in the system.

- **Amount of user data is very less**

The amount of data available for each user is often very limited in real applications. During the enrollment phase, users are often required to supply only a few samples of their signatures. In other words, even if there is a large number of users enrolled to the system, a classifier needs to perform well for a new user, for whom only a small set of samples are available.

- **Static image offers less information**

Offline signatures are nothing but handwritten signatures scanned and stored as static image. Amount of information is very less compared to online signatures. Online signatures offers dynamic features like speed, pen pressure, directions, stroke length and when the pen is lifted from the paper. These information is nowhere available in images.

- **Prone to noise**

As offline signature is scanned document, probability of occurring noise is quite high. These noise often lead to poor performance.

- **Isolation of Region of Interest**

Usually, a handwritten signature comes with a document. From that document isolating the signature portion only is a difficult task. There is no automated way to do so as ROI varies from document to document. Isolation of ROI is done manually, which is tiresome and prone to error and noise.

## 1.4 Thesis Objectives

Our research targeted to find an appropriate representation of the signature image which can be used for graph matching. Suitable feature descriptors to be selected to ensure the performance of the verification system. Our graph matching based verification approach should satisfy the following requirements:

- To propose a appropriate graph representation of signature image that contains the structural information of the signature
- To find a robust graph matching method which can provide better correspondence between signature images
- To perform a comparative analysis between the proposed method and different existing methods under different experimental setup using proper benchmark dataset.

## 1.5 Thesis Contributions

In this work, we have proposed an Offline Signature Verification method using Factorized Graph matching. The main contributions of this thesis can be summarized as follows:

- A new method of calculating the affinity matrix is proposed to preserve the structural accuracy of the graph.
- For edge feature, normalized edge direction is added along with the edge weight which preserve more structural information of the signature.
- Global Affinity Matrix is factorized so that the graph matching process converge faster.

## 1.6 Thesis Organization

The rest of the thesis will be organized as follows: in Chapter 2, we present the literature review of existing methods and their performance as well as limitations for offline signature verification system. In Chapter 3, we details of our proposed methodology is given. There, we discuss about the overall idea of our proposed method and step by step implementation process. In Chapter 4, experimental set up, experimental results and performance analysis of our proposed idea with various promising methods are discussed. Finally, in Chapter 5, we conclude our thesis contributions and shows the future scopes for further developing the proposed method.

# Chapter 2

## Literature Review

Over the years, offline signature verification has been studied from many perspectives. Most of the methods use a similar pattern for verification system. For Writer Dependent (WD) system there are two phases mainly. One is the training phase and the other one is the testing phase for verification. Architecture of the training phase is illustrated in figure 2.1. Block diagram in figure 2.2 depicts the outline of testing

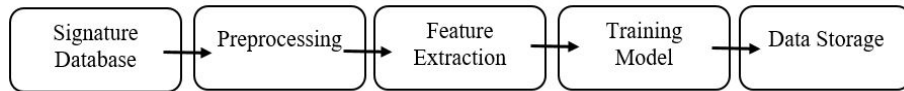


Figure 2.1: Block diagram of existing training methods

phase: During the training phase, two important steps are- Feature Extraction and

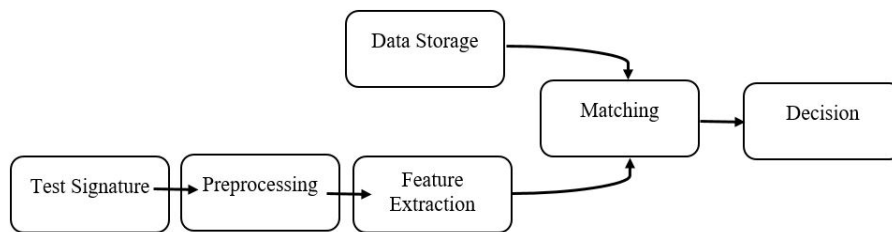


Figure 2.2: Block diagram of existing testing methods

Training Model. Most of the recent works varies based on these two steps. Feature extraction gives a way to represent the signature image whereas training model is how to differentiate between signatures. In this chapter we present existing works for offline signature verification based on these two important components of the method.

### 2.1 Feature Extraction

Studies in this regard has yielded multiple alternatives for feature extraction. Broadly speaking, the feature extraction techniques can be classified as Static or Pseudo-dynamic, where pseudo-dynamic features attempt to recover dynamic information

from the signature execution process (such as speed, pressure, etc.). Another broad categorization of the feature extraction methods is between Global and Local features. Global features describe the signature images as a whole - for example, features such as height, width of the signature, or in general feature extractors that are applied to the entire signature image. In contrast, local features describe parts of the images, either by segmenting the image (e.g. according to connected components) or most commonly by the dividing the image in a grid (of Cartesian or polar coordinates), and applying feature extractors in each part of the image.

### 2.1.1 Geometric Features

Geometric features measure the overall shape of a signature. This includes basic descriptors, such as the signature height, width, caliber (height-to-width ration) and area. More complex descriptors include the count of endpoints and closed loops [1]. Besides using global descriptors, several authors also generate local geometric features by dividing the signature in a grid and calculating features from each cell. For example, using the pixel density within grids [6], [7], [8]. A.C. Verma et al. [9] proposed a method that considers global and geometric features like aspect ratio, center of gravity, baseline shift, Slope of line obtained from curve fitting etc. Mean of each feature calculated from the training data. This mean is regarded as the prototype of a user. Variance of the input signature from the mean is calculated for decision making. For the basis of comparison metric, Euclidian distance is calculated.

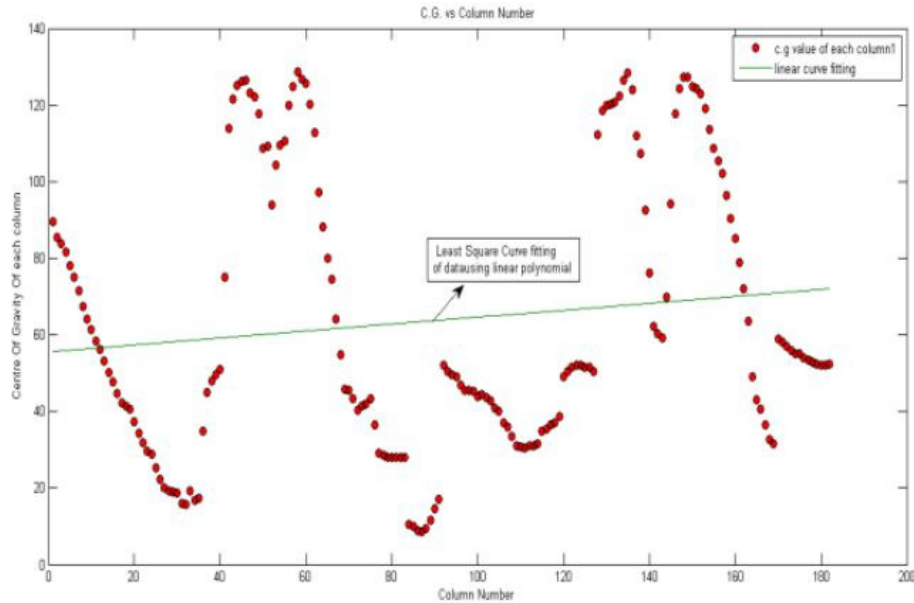


Figure 2.3: Slope of line obtained from curve fitting of centre of gravity of each column.

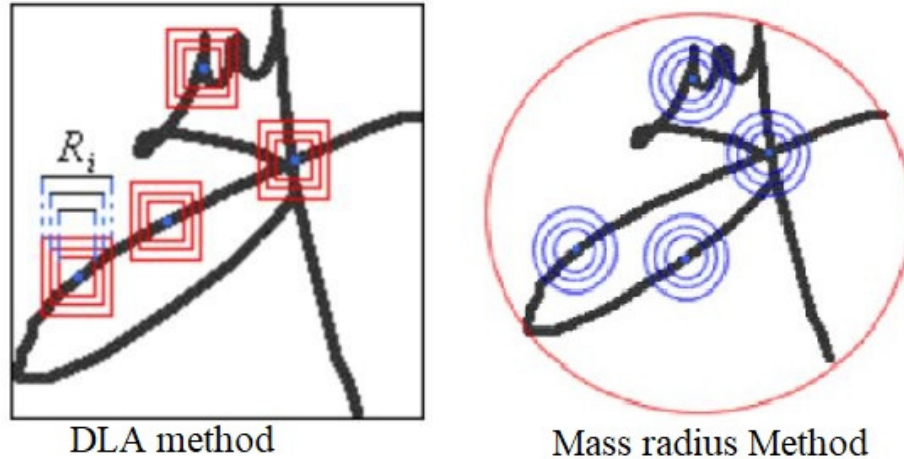


Figure 2.4: Computation of fractal dimension

### 2.1.2 Mathematical transformations

Researchers have used a variety of mathematical transformations as feature extractors. Nemcek and Lin [10] investigated the usage of a fast Hadamart transform and spectrum analysis for feature extraction. Pourshahabi et al. [11] used a Contourlet transform as feature extraction, stating that it is an appropriate tool for capturing smooth contours. Coetzer et al. [12] used the discrete Radon transform to extract sequences of observations, for a subsequent HMM training. Deng et al [13] proposed a signature verification system based on the Wavelet transform. Zouari et al [14] has investigate the usage of the Fractal transform for the problem. For each signature its Fractal Dimension FD is calculated using three methods (Box Counting, Mass Radius, DLA). These values are then stored in a vector  $V$  that will form the feature vector of the signature. From all the sample average fractal information is calculated where standard deviation is stored for future computation. Figure 2.4 shows two prominent methods for counting fractal dimension of a signature.

### 2.1.3 Texture features

Texture features, in particular variants of Local Binary Patterns (LBP), have been used in many experiments in recent years. The LBP operator describe the local patterns in the image, and the histogram of these patterns is used as a feature descriptor. LBP variations have been used in many studies [15], [16], [17], [18], and have demonstrated to be among the best hand-crafted feature extractors for this task. Another important texture descriptor is GLCM (Gray Level Co-occurrence Matrix). This feature uses relative frequencies of neighboring pixels, and was used in a few papers [18], [19].



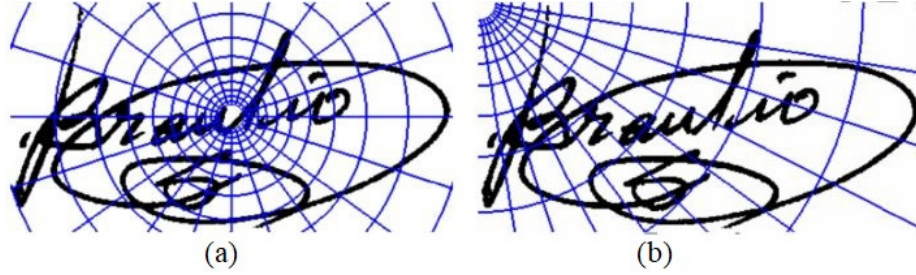


Figure 2.5: Log polar grids used in [1] varying the origin (a) Origin at the center (b) Origin at the top left corner

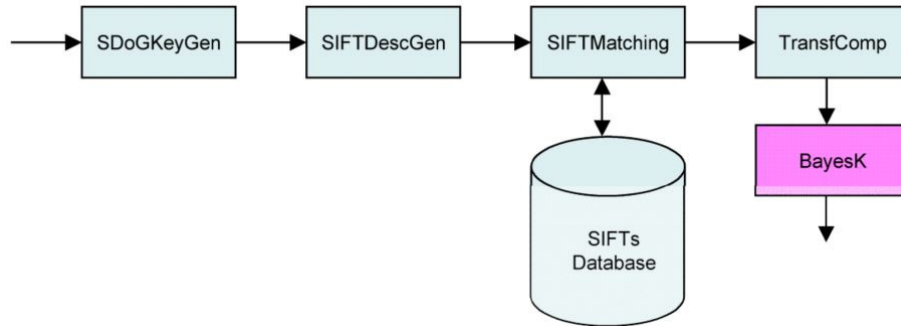


Figure 2.6: Block diagram of the proposed signature verification system. SDoGKeyGen: SDoG Keypoints Generation; SIFTDescGen: SIFT Descriptors Generation; TrasfComp: Transformation Computation and hypothesis rejection tests; BayesK: Bayes Classifier. [2]

### 2.1.4 Interest point matching

Interest point matching methods, such as SIFT (Scale-Invariant Feature Transform) and SURF (Speeded Up Robust Features) have been largely used for computer vision tasks. Ruiz-del-Solar et al. [2] used SIFT to extract local interest points from both query and reference samples to build a writer-dependent classifier. After extracting interest points from both images, they generated a set of 12 features, using information such as the number of SIFT matches between the two images, and processing time. Block diagram of this method is depicted in Figure 2.6.

Malik et al. [3] used SURF to extract interest points in the signature images, and used these features to assess the local stability of the signatures. During classification, only the stable interest points are used for matching. Stable interest points are determined by Heat maps (shown in Figure 2.7) of genuine signatures. The number of keypoints in the query image, and the number of matched keypoints were used to classify the signature as genuine or forgery.

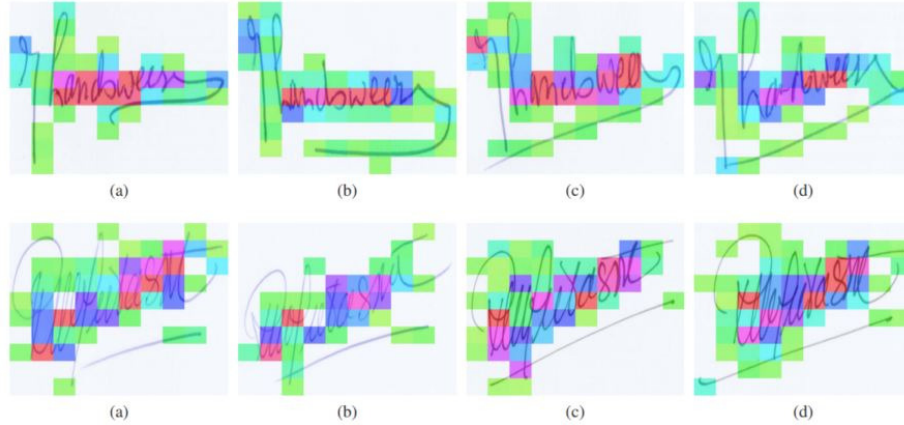


Figure 2.7: Heat maps of some example specimen (genuine) signatures from two different authors (one genuine author in each row) showing the most stable (green) and the most unstable (red) parts along with the moderately stable parts (colors varying from green to red through blue)[\[3\]](#)

### 2.1.5 Graph Matching

In graph based matching, a graph representation of the signature is created for the supplied signature image. Then graph matching algorithms are used to find point wise correspondence between two images. S. I. Abuhaiba [\[20\]](#), presented a signature verification method that depends on the raw binary pixel intensities instead of complex set of features. In this approach, signature verification problem is viewed as a graph matching problem, for which the Hungarian method was used to solve. Steps of this method is as following:

- First, signature is preprocessed as described in the Figure [2.8](#).
- Set of pixels in the final image constitute the set of vertices of the graph
- From two signature images, set of pixels X and Y combined to construct a complete bipartite graph
- To find the correspondence of this assignment problem (AP), Hungarian method is used
- Decision is taken thresholding the cost of Hungarian algorithm

This method is one of the earliest works that used graph matching in the field of offline signature verification. This method gets a moderate result for False Acceptance Rate (FAR) and False Rejection Rate (FRR) on the author’s hand-crafted dataset but fails in accuracy. One of the main reason is that only pixels does not provide enough distinction as a feature descriptor. As pixels are considered to be vertex of the graph, presence of noise will affect the result a lot.

Ghosh et al. [\[4\]](#) used weighted complete bipartite graph for signature representation and bipartite matching for verification. Steps of this method are-

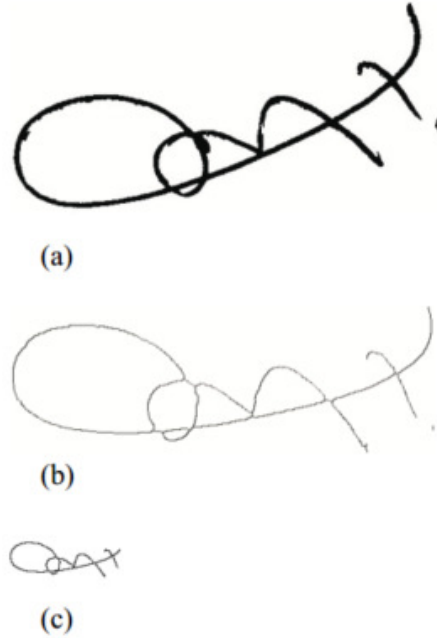


Figure 2.8: Signature preprocessing: a) original image, b) image after pepper noise removal, skew elimination, smoothing, and thinning, and c) final 64x128 normalized image.

- Image is subdivided recursively based on the center of gravity.
- COG of each sub image is considered as the set of vertices
- Geometrically closer points are connected to construct the graph
- For each of the edges, angle with the COG is computed for the feature vector by Equation 2.1 considering that the two end points of the edge form a triangle with COG as shown in Figure 2.9.

$$A = \text{COS}^{-1}((b^2 + c^2 - a^2)/(2bc)) \quad (2.1)$$

- Before classification step, a set of K sample signatures are collected from a person and the feature of the signature i.e.  $(2^{N+1} - 2)$  number of angles are extracted for each signature and stored into the database. A threshold value T is calculated from the set of K signatures and is also stored in the database.
- If more than 80 percent of the feature points are matched than the asked signature is accepted.

This method uses weighted complete bipartite graph, but fails as for handwritten signatures, COG varies for little changes in orientation. That's why it cannot handle intra-class variability of offline signature. Some authors proposed a combination of Graph matching and Cross-validation principle for signature verification. [21][22]

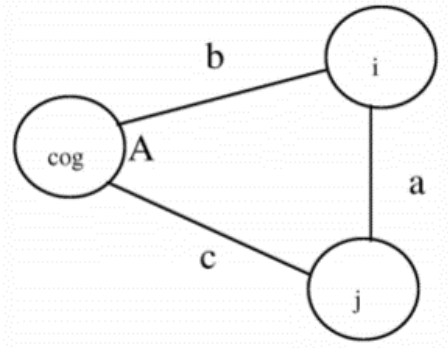


Figure 2.9: Feature computation of each edge in the Adjacency matrix [4]

Table 2.1: Speed comparison of feature detectors on signature image

Feature detection method	Time (ms)	# of keypoints	Time / keypoint
SIFT	112.98	598	0.1889
SURF	35.98	855	0.04208
FAST	8.0003	930	0.0086
ORB	9.003	475	0.01895
BRISK	81.984	1307	0.06273

In graph matching one of the important task is to create the graph representation of the signature image. Some of them use COG positions as nodes of the graph. Some use simple pixel values for it. But, a more robust approach for that is to use interest points as the nodes of the graph. There are a number of established feature detectors. Finding a suitable feature detector is very important for matching. As for our experiment of offline signature verification speed and accuracy both play important roles, a detector with good speed and accuracy is to find. Many comparative experiments have already been led to find suitable detector such as [23], [24], [25], [26]. From those analysis, it is quite clear that FAST corner detector is the fastest of them all and has a good tracking record. We also tested a number of promising feature detectors on signature images. Result of that test is depicted in Table 2.1. From the numbers, it is clear that FAST corner detection is at least twice faster than the second best ORB. Thus it makes FAST a popular choice for faster computation.

## 2.2 Training Model

The classifiers for signature verification can be broadly classified in two groups: writer dependent and writer-independent. In the first case, which is more common in the literature, a model is trained for each user, using the user’s genuine signatures, and random forgeries (by using genuine signature from other users). During the operations phase, the model trained for the claimed identity is used to classify query signatures

as genuine or forgery. The writer-independent approach, on the other hand, involves only a single classifier for all users. In this case, the system learn to compare a query signature with a reference. During the test phase, the model is used to compare a query signature with reference genuine samples from the claimed individual, to make a decision. One common way of training WI systems is to use a dissimilarity representation, where the inputs to the classifiers are differences between two feature vectors with a binary label that indicates whether the two signatures are from the same user or not [27], [28].

### 2.2.1 Hidden Markov Models

Several authors have proposed using Hidden Markov Models for the task of signature verification [8], [29], [30]. In particular, HMMs with a left-to-right topology have been mostly studied, as they match the dynamic characteristics of American and European handwriting (with hand movements from left to right).

In the work from Justino [8], Oliveira [29] and Batista [30], the signatures are divided in a grid format. Each column of the grid is used as an observation of the HMM, and features are extracted from the different cells within each column, and subsequently quantized in a codebook. In the verification phase, a sequence of feature vectors is extracted from the signature and quantized using the codebook. The HMM is then used to calculate the likelihood of the observations given the model. After calculating the likelihood, a simple threshold can be used to discriminate between genuine signatures and forgeries [8], or the likelihood itself can be used for more complex classification mechanisms [30].

### 2.2.2 Support Vector Machines

Support Vector Machines have been extensively used for signature verification, for both writer-dependent and writer independent classification [31], [32], [33], [34], [1], [35], empirically showing to be the one of the most effective classifiers for the task. In recent years, Guerbai et al [36] used One-Class SVMs for the task. This type of model attempt to only model one class (in the case of signature verification, only the genuine signatures), which is a desirable property, since for the actual users enrolled in the system we only have the genuine signatures to train the model. However, the low number of genuine signatures present an important challenge for this strategy.

### 2.2.3 Neural Networks and Deep Learning

Neural Networks have been explored for both writer-dependent and writer-independent systems. Huang and Yan [37] used Neural Networks to classify between genuine signatures and random and targeted forgeries. They trained multiple networks on features extracted at different resolutions, and another network to make a decision, based on the outputs of these networks. Shekar et al [38] presented a comparison of neural networks and support vector machines in three datasets. More recently, Soleimani et al. [39] proposed a Deep Multitask Metric Learning (DMML)

system for signature verification. In this approach, the system learns to compare two signatures, by learning a distance metric between them. The signatures are processed using a feed forward neural network, where the bottom layers are shared among all users (i.e. the same weights are used), and the last layer is specific to each individual, and specializes for the individual. In the work of Rantzsch et al. [40], a metric learning classifier is learned, jointly learning a feature representation, and a writer independent classifier.



# Chapter 3

## Proposed Method

In this chapter, we explain the proposed methodology for Offline Signature Verification using Factorized Graph Matching. The overall idea of our proposed method is depicted in Fig. 3.1. The proposed method uses the following steps: 1) Pre-processing, 2) Keypoint Detection, 3) Graph Formulation, 4) Feature Extraction, 5) Affinity Matrix Computation, 6) Factorized Graph Matching, and 7) Decision Making.

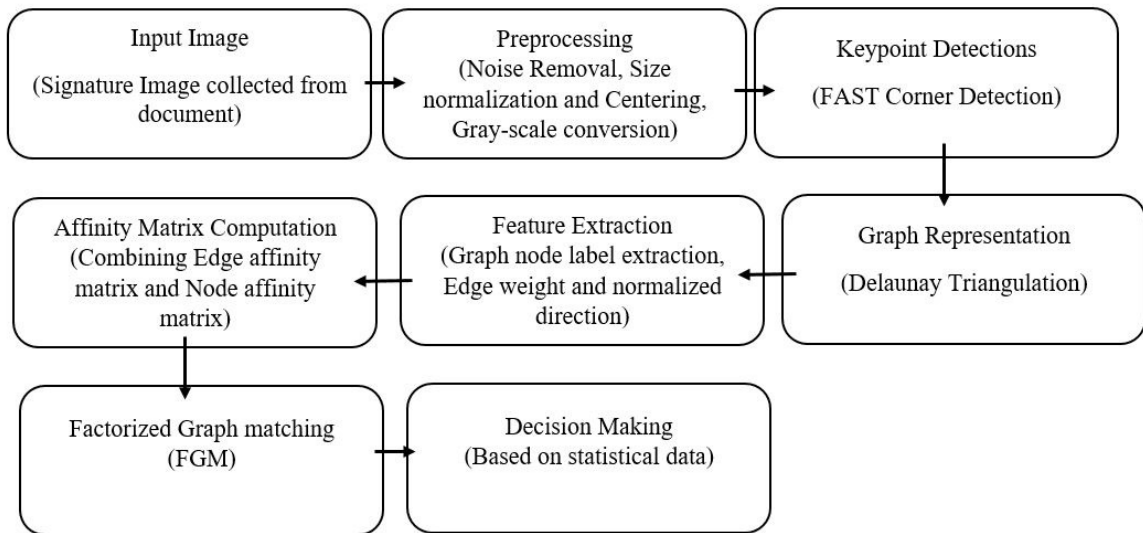


Figure 3.1: Steps of the proposed system

### 3.1 Pre-processing

As with most pattern recognition problems, preprocessing plays an important role in signature verification. Signature images may present variations in terms of pen thickness, scale, rotation, etc., even among authentic signatures of a person. Bellow we summarize the main pre-processing techniques:

### 3.1.1 Size normalization and centering

The whole signature is not important for the operation. Region of interest is extracted using cropping the original image. The simplest strategy is to crop the signature images to have a tight box on the signature. Cropping is done with respect to bounding box of the supplied signature by calculating the first foreground row, first foreground column, last foreground row and last foreground column.

### 3.1.2 Gray-scale Conversion

Our proposed method work best with the gray scale images. But some signature may be given as color image. In that case for the better performance of the verification process we first convert the given color image to the gray scale image. For that conversion we just follow the following conversion equation:

$$I = 0.299R + 0.587G + 0.114B \quad (3.1)$$

### 3.1.3 Noise Removal

Removal of random noises is one of the important steps in pre-processing. Scanned signature images often contain noise. Simple Filters is used for the removal of random noises from the supplied signature. Considering the practical scenario supplied signature may contain a number of different noises; which are reduced in the pre-processing step. For example the following left image of signature shows the presence of ‘salt and pepper’ which has been removed in the right one by applying Median filter.

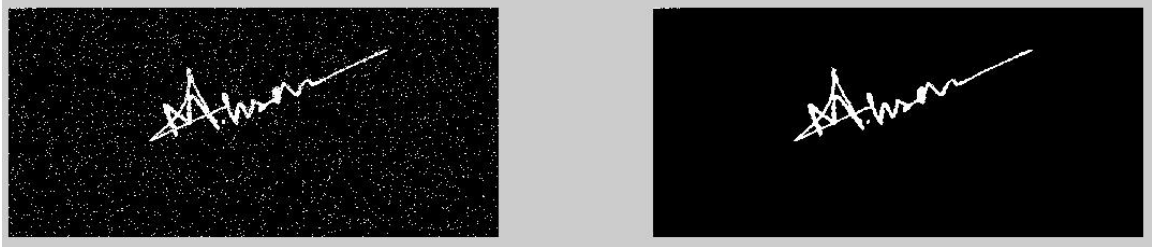


Figure 3.2: Effect of noise removal from signature Image

## 3.2 Keypoint Detection

A Keypoint is a point in an image which has a well-defined position and can be robustly detected. This means that an interest point can be a corner but can also be, for example, an isolated point of local intensity maximum or minimum, line endings, or a point on a curve where the curvature is locally maximum. According to the discussion in section 2.1.5 and Table 2.1, we choose FAST corner detection to detect keypoints.

FAST corner detection algorithm was presented on 2010 by Rosten et al. [41] It is

proposed based on SUSAN corner criterion [42]. Similar to SUSAN, FAST takes a circle of 16 pixels from the neighborhood of a potential keypoint candidate. These 16 pixels can be chosen by a Bresenham circle [43] of radius 3. Based on the value of these pixels, it is determined whether the candidate is a keypoint or not. As plotted in figure x, the intensity values of the chosen pixels are compared with the pixel of the candidate. If  $n$  pixels among the 16 fulfills the threshold criterion then the candidate is taken as interest point the value of  $n$  is usually taken as 12. To make it faster, all 16 pixels are indexed clockwise and FAST compares the intensity values of pixels 1, 5, 9, 13 from the circle. At least any three of these four pixels should be within the threshold for the candidate to be keypoint. If a candidate pass this test, FAST goes for further testing. Otherwise it rejects the candidate. This faster approach works with good speed but has a few weakness. Along with the other weakness, numbering the pixels order is an overhead and multiple features are detected adjacent to one another. To overcome those weakness machine learning approach is taken and the keypoints being adjacent to each other is addressed using non-maximal suppression. Noting that Figure 3.3 is taken from website in [44]. So, in

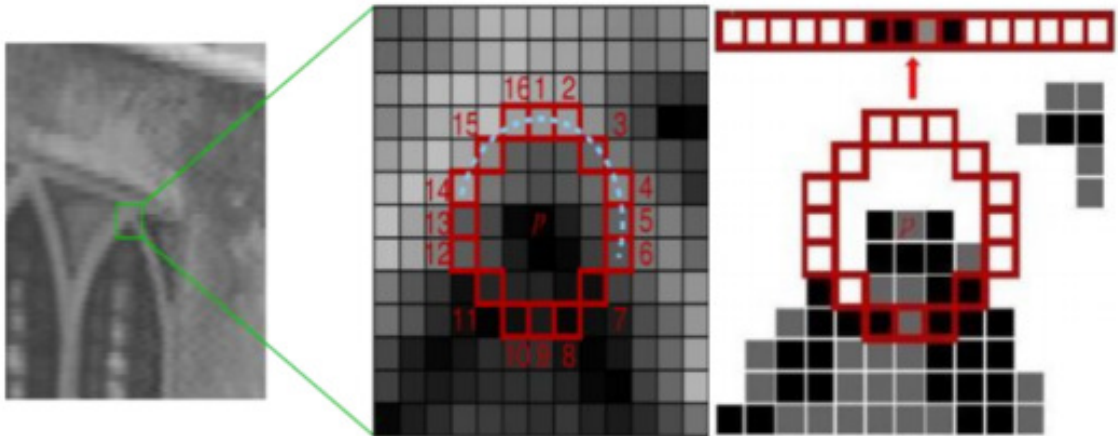


Figure 3.3: (a) A processed interest point and 16 pixels surrounding on it, (b) the demonstration of storing 16 values surrounding pixels in a vector form.

this step of our proposed method we find out the key-point of the supplied signature using FAST Corner Detection Algorithm. Those key-points define the exact skeleton of the supplied signature. Figure 3.4 and Figure 3.5 shows the output of keypoint detection on Chinese and Dutch Dataset respectively.

### 3.3 Graph Formulation

The next step is to create graph representation of the given signature. As we already have the keypoints depicting the skeleton of the signature image, we can easily generate planar graph for further computation. There are many established methods for generating graph from keypoints. Most famous and efficient of them is Delaunay Triangulation. [45]



Figure 3.4: Output of Keypoint detection on Chinese Dataset

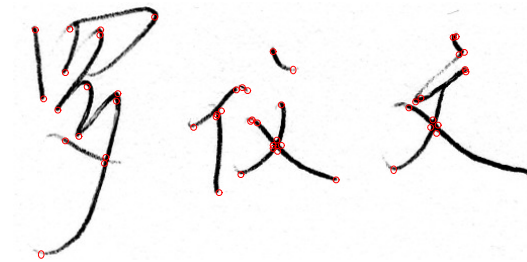


Figure 3.5: Output of Keypoint detection on Chinese Dataset

Given a set of points in a plane, a triangulation refers to the subdivision of the plane into triangles, with the points as vertices. In Figure 3.7, we see a set of landmarks on the left image, and the triangulation in the middle image. A set of points can have many possible triangulations, but Delaunay triangulation stands out because it has some nice properties. In a Delaunay triangulation, triangles are chosen such that no point is inside the circumcircle of any triangle. Figure 3.6 shows Delaunay triangulation of 4 points A, B, C and D. In the top image, for the triangulation to be a valid Delaunay triangulation, point C should be outside the circumcircle of triangle ABD, and point A should be outside the circumcircle of triangle BCD. An interesting property of Delaunay triangulation is that it does not favor “skinny” triangles ( i.e. triangles with one large angle ).

Figure 3.6 shows how the triangulation changes to pick “fat” triangles when the points are moved. In the top image, the points B and D have their x-coordinates at  $x = 0$ , and in the bottom image they are moved to the right to  $x = 0.2$ . In the top image angles ABC and ABD are large, and Delaunay triangulation creates an edge between B and D splitting the two large angles into smaller angles ABD, ADB, CDB, and CBD. On the other hand in the bottom image, the angle BCD is too large, and Delaunay triangulation creates an edge AC to divide the large angle.

We have used this triangulation method to generate a planar graph which represents the signature. The result of this operation is illustrated in Figure 3.7

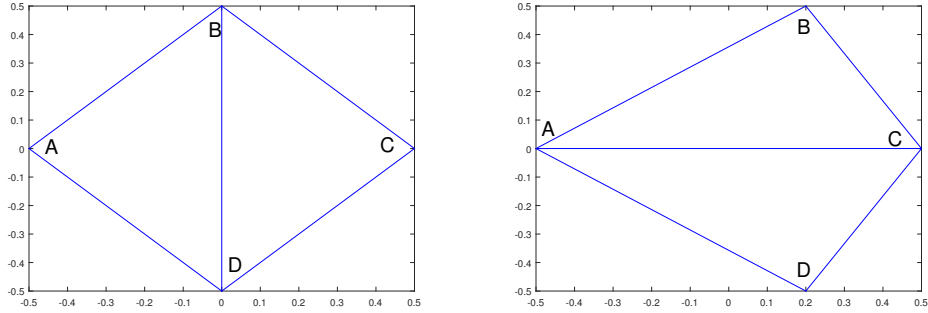


Figure 3.6: Delaunay Triangulation does not favor skinny triangles

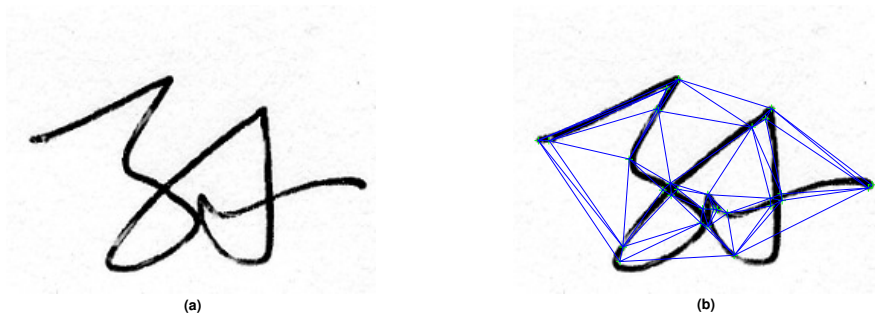


Figure 3.7: Graph Formulation from keypoint, (a) Original image (b) Formulated graph

### 3.4 Feature Extraction

For matching two signatures an appropriate and distinctive descriptor of the graphs needs to be computed. We denote the graph as  $G = \{V, E\}$  where  $V$  is the set of all the nodes and  $E$  is the set of edges of the graph. Nodes are keypoints which preserve the point wise correspondence while edge features provide structural information of the graph. Features descriptors of the keypoints can be used as node label. There are a number of popular methods for computing keypoint features. Performance of those descriptors have been analyzed in multiple surveys like [46], [26], [24]. Based on those surveys, interest point based feature descriptors can be categorized under two classes.

1. **Histogram of Gaussian (HoG) Based descriptors**

Histograms of this class are prominent for their performance in accuracy. They calculate the features in float points. Members of this family are- SIFT, SURF and GLOH. The most popular member of them is SIFT. SURF is close competitor to SIFT in performance but superior in speed with a significant margin.

2. **Binary descriptors**

Binary descriptors encode the information of a patch in binary strings. It compares the intensity of the reference points in the patches and stores the result in

either 0 or 1. These operations are fast and can be stored using very less amount of space. A Distance measure between two binary strings is computed using Hamming distance. Then matching of binary strings can be done using a single XOR operation of the processor. These are the exact motivation behind the binary descriptors. Most of the binary descriptors work in similar fashion with small differences.

According to many of the researchers, SURF feature descriptor is the most suitable one for image matching. Along with FAST as feature detector, SURF provides the best results in similar applications. Bayraktar et al. [24] is one of them. In this step we will compute the SURF features [47] for the keypoints which are nodes in the graph. Given the location of the keypoints, for feature descriptor of each keypoints SURF uses wavelet responses in horizontal and vertical direction for a neighborhood of size  $6s$ . Adequate Gaussian weights are also applied to it. Then they are plotted in a space as given in Figure 3.8. The dominant orientation is estimated by calculating the sum of all responses within a sliding orientation window of angle 60 degrees. Interesting thing is that, wavelet response can be found out using integral images very easily at any scale. SURF provides such a functionality called Upright-SURF or U-SURF. It improves speed and is robust upto  $\pm 15^\circ$ .

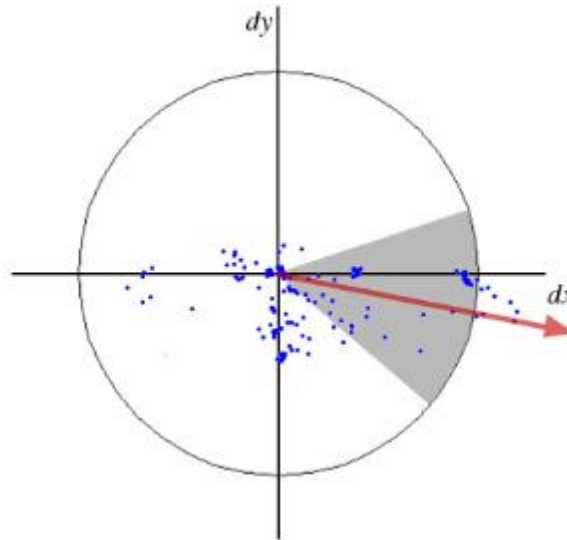


Figure 3.8: Wavelet response of keypoint neighborhood

For feature description, SURF uses Wavelet responses in horizontal and vertical direction (again, use of integral images makes things easier). A neighborhood of size  $20s \times 20s$  is taken around the keypoint where  $s$  is the size. It is divided into  $4 \times 4$  subregions. For each subregion, horizontal and vertical wavelet responses are taken and a vector is formed like this,  $v = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$ . This when represented as a vector gives SURF feature descriptor with total 64 dimensions. Lower the dimension, higher the speed of computation and matching, but provide better distinctiveness of features.

For more distinctiveness, SURF feature descriptor has an extended 128 dimension

version. The sums of  $d_x$  and  $|d_x|$  are computed separately for  $d_y \leq 0$  and  $d_y \geq 0$ . Similarly, the sums of  $d_y$  and  $|d_y|$  are split up according to the sign of  $d_x$ , thereby doubling the number of features. It does not add much computation complexity.

In our method, we propose to use SURF features of 128 dimension as it provides more distinctiveness.

For the features of edge, the edge weight is a popular metric to use. [48] The length for each of the edges existing in the graph is computed and normalized by the resolution of the signature image. It is then stored as a feature for further computation. To provide more distinctiveness, we propose to use the direction of the edges as an important feature. Only the edge weight does not perform well for verification. We calculated the orientation of each edges for further use.

More specifically, each edge is represented by a couple of values,  $[q_c, \theta_c]$ , where  $q_c$  is the pairwise distance between the connected nodes and  $\theta_c$  is the angle between the edge and the horizontal line.

### 3.5 Affinity Matrix Computation

We denote a graph by  $\mathfrak{S} = \{P, Q, R, G\}$ , where  $P = [p_1, \dots, p_n] \in \mathbb{R}^{dp \times n}$ ,  $Q = [q_1, \dots, q_m] \in \mathbb{R}^{dq \times m}$  and  $\theta = [\theta_1, \dots, \theta_n] \in \mathbb{R}^{dp \times n}$  are the feature matrices computed for nodes, edge weight and edge direction respectively. The topology of  $\mathfrak{S}$  is specified by a node-edge incidence matrix  $G \in \{0, 1\}^{n \times m}$ , where  $g_{ic} = g_{jc} = 1$  if the  $i^{th}$  and  $j^{th}$  nodes are connected by the  $c^{th}$  edge, and zero otherwise.

Suppose that we are given a pair of graphs,  $\mathfrak{S} = \{P1, Q1, \theta1, G1\}$  and  $\mathfrak{S} = \{P2, Q2, \theta2, G2\}$ . We compute three affinity matrices,  $K_p \in \mathbb{R}^{n1 \times n2}$ ,  $K_q \in \mathbb{R}^{m1 \times m2}$  and  $K_\theta \in \mathbb{R}^{m1 \times m2}$ , for measuring the similarity of each node and edge pair respectively. Computation of node label  $p_c$ , edge weight  $q_c$  and edge direction  $\theta_c$  is illustrated in previous sections. These three affinity matrices are computed according to the following equations:

$$K_{c_1 c_2}^p = \exp(-|p_{c_1} - p_{c_2}|) \quad (3.2)$$

$$K_{c_1 c_2}^q = \exp(-(q_{c_1} - q_{c_2})^2) \quad (3.3)$$

$$K_{c_1 c_2}^r = \exp(-(\theta_{c_1} - \theta_{c_2})^2) \quad (3.4)$$

More specifically,  $k_{i_1 i_2}^p = \phi_p(p_{i_1}^1, p_{i_2}^2)$  measures the similarity between the  $i_1^{th}$  node of  $\mathfrak{S}_1$  and the  $i_2^{th}$  node of  $\mathfrak{S}_2$ , and  $k_{c_1 c_2}^q = \phi_q(q_{c_1}^1, q_{c_2}^2)$  measures the similarity between the  $c_1^{th}$  edge of  $\mathfrak{S}_1$  and the  $c_2^{th}$  edge of  $\mathfrak{S}_2$ . The problem of graph matching consists in finding a correspondence between the nodes of  $\mathfrak{S}_1$  and  $\mathfrak{S}_2$  that maximizes the following score of global consistency:

$$J_{gm}(X) = \sum_{i_1 i_2} x_{i_1 i_2} k_{i_1 i_2}^p + \sum_{\substack{i_1 \neq i_2, j_1 \neq j_2 \\ g_{i_1 c_1}^1 g_{j_1 c_1}^1 = 1 \\ g_{i_2 c_2}^1 g_{j_2 c_2}^1 = 1}} x_{i_1 i_2} x_{j_1 j_2} k_{c_1 c_2}^q + \sum_{\substack{i_1 \neq i_2, j_1 \neq j_2 \\ g_{i_1 c_1}^1 g_{j_1 c_1}^1 = 1 \\ g_{i_2 c_2}^1 g_{j_2 c_2}^1 = 1}} x_{i_1 i_2} x_{j_1 j_2} k_{c_1 c_2}^\theta \quad (3.5)$$



Where  $X \in \{0, 1\}^{n_1 \times n_2}$  denotes the node correspondence, i.e.,  $x_{i_1 i_2} = 1$  if the  $i_1^{th}$  node of  $\mathfrak{S}_1$  corresponds to the  $i_2^{th}$  node of  $\mathfrak{S}_2$ . In most cases,  $X$  is constrained to be a one-to-one matching, i.e.,  $X \mathbf{1}_{n_2} \leq \mathbf{1}_{n_1}$  and  $X^T \mathbf{1}_{n_1} \leq \mathbf{1}_{n_2}$ .

It is more convenient to write  $J_{gm}(X)$  in a quadratic form,  $x^T K x$ , where  $x = \text{vec}(X) \in \{0, 1\}^{n_1 n_2}$  is an indicator vector and  $K \in \mathbb{R}^{n_1 n_2 \times n_1 n_2}$  is computed as follows:

$$K_{i_1 i_2 j_1 j_2} = \begin{cases} k_{i_1 i_2}^p, & \text{if } i_1 = j_1 \text{ and } i_2 = j_2 \\ \frac{1}{2} k_{c_1 c_2}^q + \frac{1}{2} k_{c_1 c_2}^\theta, & \text{if } i_1 \neq j_1 \text{ and } i_2 \neq j_2 \text{ and } g_{i_1 c_1}^1 g_{j_1 c_1}^1 g_{i_2 c_2}^1 g_{j_2 c_2}^1 = 1 \\ 0, & \text{otherwise} \end{cases} \quad (3.6)$$

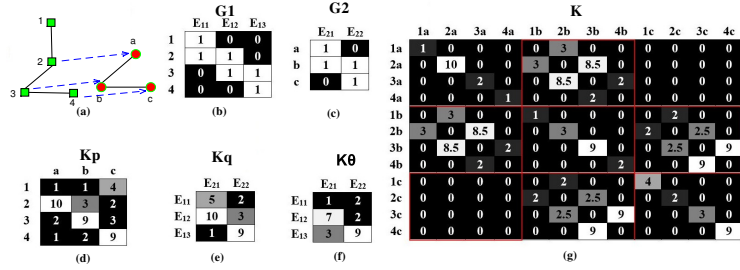


Figure 3.9: Example of graph matching and related matrices. (a) Two synthetic graphs. (b) The 1<sup>st</sup> graph's incidence matrix G1. (c) The 2<sup>nd</sup> graph's incidence matrix G2. (d) The node affinity matrix Kp. (e) The edge distance affinity matrix Kq. (f) The edge direction affinity matrix Kq. (g) The global affinity matrix K

Figure 3.9 describes the method of computing the global affinity matrix K from node and edge affinity matrices.

### 3.6 Factorized Graph Matching (FGM)

After computing the affinity matrix K, the goal of graph matching is to optimize the following QAP:

$$\max_x x^T K x, \quad \text{s. t. } Ax \leq b \text{ and } x \in \{0, 1\}^{n_1 n_2}, \quad (3.7)$$

where  $A = \begin{bmatrix} \mathbf{1}_{n_2}^T & \otimes \mathbf{I}_{n_1} \\ \mathbf{1}_{n_2} & \otimes \mathbf{I}_{n_1}^T \end{bmatrix}$  and  $b = \mathbf{1}_{n_1 + n_2}$ .

It is well known that the QAP is one of the most difficult combinatorial optimization problems. In general, instances of size  $n \geq 20$  cannot be exactly solved in practical time. Many methods have been proposed to compute an approximate solution. In particular, most efforts focus on maximizing  $J_{gm}(X)$  by relaxing the binary constraints. For instance, a popular relaxation is to constrain  $X$  as a doubly stochastic



matrix, [49] [50] [51] [52] which is the convex hull of permutation matrices. Though the constraint can be relaxed to be convex, we still need to tackle a hard non-convex quadratic programming since  $K$  is not necessarily negative definite.

To be able to derive a better optimization scheme for addressing the non-convex issue, this section exploits the underlying structure of  $K$ . In particular,  $K$  can be factorized into smaller matrices. With this new factorization of  $K$ , many graph matching methods can be re-interpreted in a coherent manner. Our main intuition relies on two observations. First, the large affinity matrix,  $K \in \mathbb{R}^{n_1 n_2 \times m_1 m_2}$  is divided into  $n_2$ -by- $n_2$  smaller blocks  $K_{ij} \in \mathbb{R}^{n_1 \times n_1}$ . Some of  $K_{ij}$ s contain only zero-value elements and their positions are indexed by  $G_2 G_2^T$ , i.e.,  $K_{ij} = 0^{n_1 \times n_1}$  if  $[G_2 G_2^T]_{ij} = 0$ . Second, all the non-diagonal elements of  $K_{ij}$  can be computed as  $G_1 \text{diag}(k_c^q) G_1^T$ .

Based on these two observations, and after some linear algebra, it can be shown that  $K$  can be factorized as:

$$K = (H_2 \otimes H_1) \text{diag}(\text{vec}(L))(H_2 \otimes H_1)^T, \quad (3.8)$$

where  $H_1 = [G_1, I_{n_1}] \in \{0, 1\}^{n_1 \times (m_1 + n_1)}$ ,

$H_2 = [G_2, I_{n_2}] \in \{0, 1\}^{n_2 \times (m_2 + n_2)}$ ,

$$L = \begin{bmatrix} K_q & -K_q G_2^T \\ -G_1 K_q & G_1 K_q G_2^T + K_p \end{bmatrix} \in \mathbb{R}^{(m_1 + n_2) \times (m_2 + n_2)},$$

Due to its combinatorial nature, this equation is usually approached by a two-step scheme:

1. solving a contiguously relaxed problem and
2. rounding the approximate solution to a binary one.

Conventional methods perform these two steps independently. As mentioned in [53] [48], however, this kind of separate treatment will inevitably cause accuracy loss, especially in the rounding step which is independent of the cost function. Inspired by [54] [48], we address these two issues in a coherent manner by iteratively optimizing an interpolation of two relaxations. This new scheme has three theoretical advantages:

1. The optimization performance is initialization-free;
2. The final solution is guaranteed to converge at an integer one and therefore no rounding step is needed;
3. The iteratively updating procedure resembles the idea of numerical continuation methods [55], which have been successfully used for solving nonlinear systems of equations in decades.

In this section, we describe a path-following strategy for optimizing Equation [3.7]. Inspired by [48], we approach the non-convex QP by iteratively optimizing a series of the following sub-problems:

$$\begin{aligned} \max_x J_\alpha(X) &= (1 - \alpha) J_{\text{vex}}(X) + \alpha J_{\text{cav}}(X), \\ \text{s.t. } X 1_{n_2} &\leq 1_{n_1}, X^T 1_{n_2} \leq 1_{n_2}, X \geq 0_{n_1 \times n_2} \end{aligned} \quad (3.9)$$

where  $\alpha \in [0, 1]$  is a tradeoff between the convex relaxation  $J_{vex}(X)$  and the concave one  $J_{cav}(X)$ . When  $\alpha = 0$ , the problem is a convex optimization problem which has a global optimal solution no matter the choice of the initialization. When  $\alpha = 1$ , the problem is a concave optimization problem which always leads to an integer solution [56] [57]. The process starts with  $\alpha = 0$  and successively increasing  $\alpha$  until 1. For a specific  $\alpha$ , we optimize  $J_\alpha(X)$  taking the Frank-Wolfe's algorithm (FW) [58], [53], [54], a simple yet powerful method for nonlinear programming. FW successively update the solution as  $X^* = X_0 + \lambda Y$  given an initial  $X_0$ . At each step, it needs to compute two components:

1. the optimal direction  $Y \in \mathbb{R}^{n_1 \times n_2}$  and
2. the optimal step size  $\lambda \in [0, 1]$ .

To compute  $Y$ , we solve linear programming using the Hungarian algorithm. Although the FW algorithm is easy to implement, it converges sub-linearly. To get faster convergence speed while keeping its advantages in efficiency and low memory cost, we adopt a modified Frank-Wolfe (MFW) [59] to find a better searching direction  $Y$  by a convex combination of previously obtained solutions.

Algorithm 1 summarizes the workflow of the algorithm. The initial  $X$  can be an arbitrary doubly stochastic matrix.

After applying FGM for a pair of signatures, we get a point wise correspondence

---

**Algorithm 1** Factorized Graph Matching (FGM)

---

input :  $K_p, K_q, K_\theta, G_1, G_2, \delta, \eta$

output:  $X$  (Correspondence Matrix)

- 1: Initialize  $X$  to be doubly stochastic matrix;
  - 2: Factorize  $L = UV_T$  with *SVD*
  - 3: **for**  $\alpha = 0 : \delta : 1$  **do**
  - 4: Path Following
  - 5: **if**  $\alpha \leq \eta$  **then**
  - 6: Optimize Equation 3.9 via CCCP to obtain  $X^*$
  - 7: **else**
  - 8: Optimize Equation 3.9 via MFW to obtain  $X^*$
  - 9: **end if**
  - 10: **if**  $J_{gm}(X^*) \leq J_{gm}(X)$  **then**
  - 11: Optimize Equation 3.7 via one step of FW to obtain  $X^*$
  - 12: **end if**
  - 13: **end for**
  - 14: Update  $X \leftarrow X^*$
- 

and objective score signifying the similarity between to signatures.

### 3.7 Signature Data Storage

For training the method, each pair of the genuine signature is compared to each other using FGM and objective scores ( $OS_1 \dots OS_m$ ) are being stored. Then statistical approach is applied proposed by Rahman et al. [60]. Mean ( $\mu_x$ ) and standard deviation ( $\sigma_x$ ) of this data are computed using equation x and stored for further use.

$$\mu_x = \frac{OS_1 + OS_2 + OS_3 + \dots + OS_m}{m} \tag{3.10}$$

$$\sigma_x = \frac{\sum(OS_i - \mu_x)^2}{m} \tag{3.11}$$

After FGM provides point wise correspondence between nodes of two genuine signature, deviation of edge direction is computed for each of the edge pairs. An average of all the deviation ( $\delta$ ) is stored. For each writer in reference signature we have collection of genuine signature, mean and standard deviation of the objective scores provided by FGM.

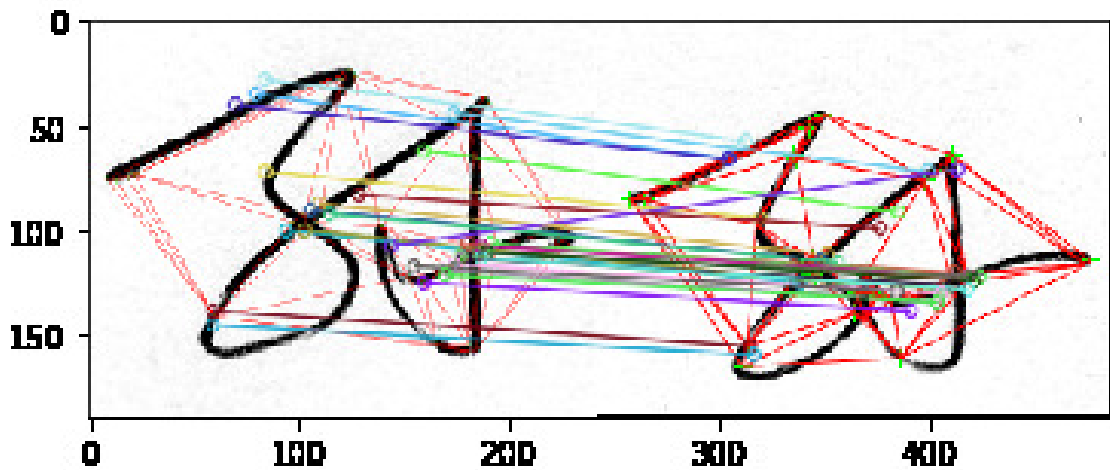


Figure 3.10: Point wise correspondence in Chinese Dataset

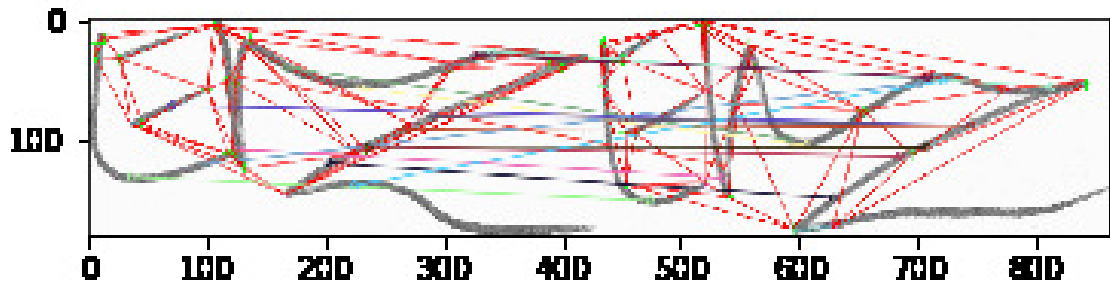


Figure 3.11: Point wise correspondence in Dutch Dataset

### 3.8 Decision Making

For verification, a test signature is provided as an input to the system. After completing necessary pre-processing steps, all the feature extraction techniques applied. For computing the affinity matrix  $K_\theta$ , small relaxation is used to match with intra-class variability. The formula for computing  $K_\theta$  is changed as following:

$$K_{c_1c_2}^r = \exp\left(-\frac{(\theta_{c_1} - \theta_{c_2})^2}{\delta}\right) \quad (3.12)$$

Once done with feature generation this signature is provided for FGM using the genuine signatures as reference. This test signature gives n objective score for n reference signature. An average ( $\mu_n$ ) of these values are computed using Equation 3.10. If ( $\mu_n$ ) is within the range then the test signature is accepted as genuine. This acceptance process is illustrated in Equation 3.13.

$$\begin{cases} \textit{Accept}, & \text{if } (\mu_x - \mu_n) \leq \sigma_x \\ \textit{Reject}, & \text{otherwise} \end{cases} \quad (3.13)$$

# Chapter 4

## Experimental Analysis

In this chapter, we evaluate the performance of the proposed methodology for offline signature verification. We will present our proposed method’s performance drawing the comparison with some other prominent methods which will be applied in two established publicly available database. Besides, in the latter part, we will also show the performance of our proposed method in case of changing other parameters and the computation time.

### 4.1 Data Set and Experimental Setup

We have implemented the proposed idea using OpenCV2 for Python by Intel Core i7 @3.20 GHz processor containing 16 GB RAM. In order to quantify the algorithmic performance of the proposed method on a signature image, the resulting segmentation is compared to its corresponding gold-standard image. There is a number of signature datasets available to test the performance of the verification system. In our experiments, we have chosen a very well-known dataset provided for ICDAR 2011 Signature Verification Competition. [\[61\]](#)

The collection contains simultaneously acquired online and offline samples. The offline dataset comprises PNG images, scanned at 400 dpi, RGB color. The online dataset comprises ASCII files with the format: X, Y, Z (per line).

#### Data Acquisition Details

A preprinted paper was used with 12 numbered boxes (width 59mm, height 23mm). The preprinted paper was placed underneath the blank writing paper. Four extra blank pages were added underneath the first two pages to ascertain a soft writing surface.

- Sampling rate 200 Hz, resolution 2000 lines/cm, precision of 0.25 mm.
- Collection device: WACOM Intuos3 A3 Wide USB Pen Tablet
- Collection software: MovAlyzer.

This dataset is divided into two subsets. One with the Dutch samples and the other with chinese samples. Details of those datasets are given below:

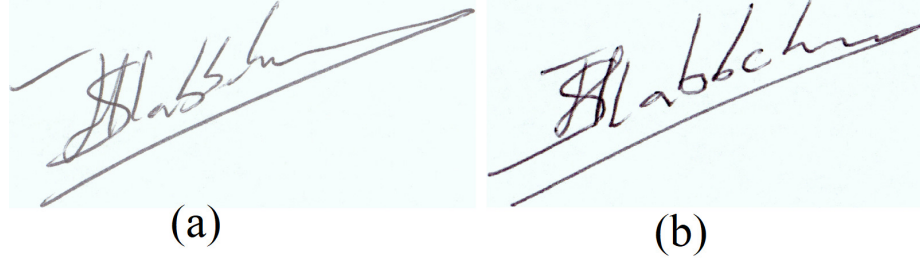


Figure 4.1: Sample signature Images from Dutch Dataset; (a) Genuine Signature, (b) Forged sample

- **Dutch dataset:**

- Training set
  - \* For both online and offline modes, signatures of 10 reference writers and skilled forgeries of these signatures.
  - \* Total online: 449 signatures, total offline: 362 signatures.
- Test Set
  - \* For both online and offline modes, signatures of 54 reference writers and skilled forgeries of these signatures.
  - \* Total online: 1907 signatures, total offline: 1932 signatures.

- **Chinese dataset:**

- Training set
  - \* For both online and offline modes, signatures of 10 reference writers and skilled forgeries of these signatures.
  - \* Total online: 659 signatures, total offline: 575 signatures.
- Test Set
  - \* For both online and offline modes, signatures of 10 reference writers and skilled forgeries of these signatures.
  - \* Total online: 680 signatures, total offline: 602 signatures.

Figure 4.1 shows genuine and forged samples from Dutch dataset. Genuine signatures are named according to the following convention (the same for all data sets): NN\_III.\*, where NN is an index of the signature and III is the ID of the reference writer, i.e., it is the NNth authentic signature contributed by writer III.

Figure 4.2 shows genuine and forged samples from Chinese dataset. Simulated signatures (forgeries) are named according to the following conventions: NN\_FFFFIII.\*, where NN is an index, FFFF is the ID of the forger, and III is the ID of the reference writer, i.e., it is the NNth simulation attempt of forger FFFF to simulate the signature of writer III. The overall summary of aforementioned two datasets is represented in table 4.1.

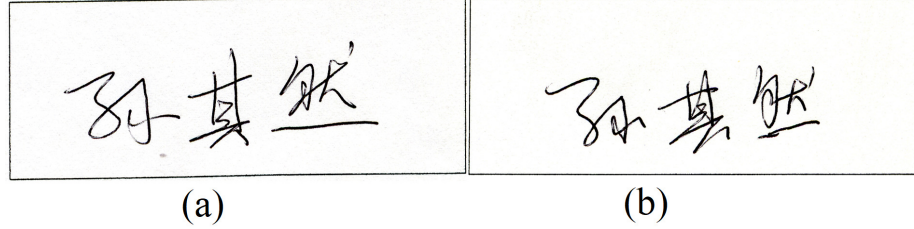


Figure 4.2: Sample signature Images from Chinese Dataset; (a) Genuine Signature, (b) Forged sample

Table 4.1: Summary of DUTCH and CHINESE dataset

Image Information	Dutch	Chinese
Number of Reference Writers	10	10
Number of Signature Images(training)	362	575
Number of Signature Images (test)	1932	602
Image format	PNG	PNG
Color Format	RGB	RGB

## 4.2 Performance Criterion

A signature verification system verifies a signature image which claims to belong to an individual. This verification process has two results. Either the signature is genuine or it is a forged one. Given a classifier and a signature, there are four possible outcomes. If the signature is genuine and it is classified as genuine, it is counted as a true positive; if it is classified as forged, it is counted as a false negative. If the signature is forged and it is classified as forged, it is counted as a true negative; if it is classified as genuine, it is counted as a false positive.

In a binary classification test, there can be four types of events. They are-

1. True Positive (TP): correctly identified
2. True Negative (TN): correctly rejected
3. False Positive (FP): incorrectly identified
4. False Negative (FN): incorrectly rejected

Given a classifier and a set of instances (the test set), a two-by-two confusion matrix (also called a contingency table) can be constructed representing the dispositions of the set of instances. This matrix forms the basis for many common metrics. Figure 4.3 shows a confusion matrix.

From this matrix many performance evaluation metrics can be derived. [62] The true positive rate (tp rate) is also known as hit rate or recall of a classifier whereas false positive rate (fp rate) is known as false alarm rate. Positive predictive value is widely known to be Precision. Another important metric is F1-measure. These

		Predicted class	
		<i>P</i>	<i>N</i>
Actual Class	<i>P</i>	True Positives (TP)	False Negatives (FN)
	<i>N</i>	False Positives (FP)	True Negatives (TN)

Figure 4.3: Confusion Matrix for signature verification

metrics can be estimated using the following equations:

$$FalseAcceptanceRate(FAR) = \frac{FP}{FP + TN} \quad (4.1)$$

$$FalseAcceptanceRate(FAR) = \frac{FN}{FN + TP} \quad (4.2)$$

$$Precision, Pr = \frac{TP}{TP + FP} \quad (4.3)$$

$$Recall, Rc = \frac{TP}{TP + FN} \quad (4.4)$$

$$Accuracy, Acc = \frac{TP + TN}{TP + TN + FP + FN} \quad (4.5)$$

$$F - Measure = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} \quad (4.6)$$

### 4.3 Performance Method Evaluation

In our proposed method, we introduced normalized direction information along with factorized graph matching. Angular information is important for signature verification as genuine signatures can have rotation variability. Again to cope up with the intra-class variability of reference images, direction should be relaxed to reduce false rejection rate. Figure 4.4 and 4.5 shows the comparison of this two methods (one with direction information, other without direction information).

As we are using a path following strategy for optimization of the QAP, convex relaxation always has a global optimal solution irrespective of the initialization. Putting extra information like angular deviation will not affect the search for global optima. For the affinity matrix factorization will lead to faster operation. By the analysis of Figure 4.4 and 4.5, we can see that adding direction information improves perfor-



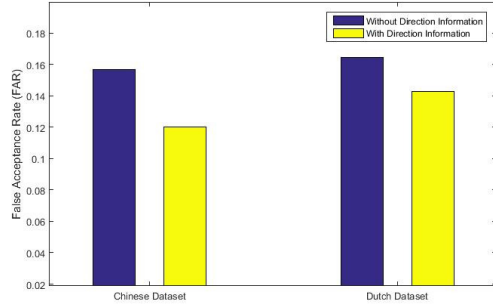


Figure 4.4: : Comparison of FAR between two proposed methods

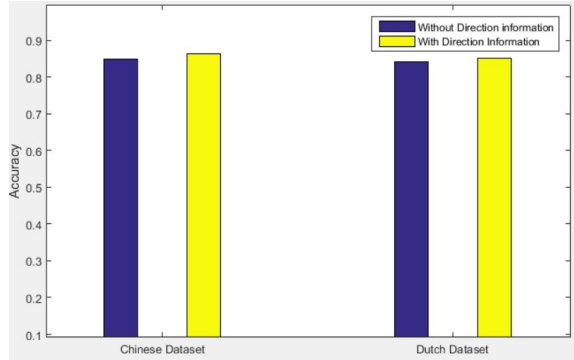


Figure 4.5: : Comparison of Accuracy between two proposed methods

mance quite significantly. Specially for reducing FAR, angular direction information plays an important role.

### 4.3.1 Comparative Analysis

For comparative analysis we compared our method with multiple promising existing works. We have chosen a wide variety of algorithms from different categories to make the comparison universal. Some evaluation results are available to corresponding author’s website whereas some results are obtained by manual implementation or collected from the original research work paper. Algorithms are listed in Table 4.2.

We have run all the methods in a single experimental setup mentioned earlier. From the outcome we compiled the result regarding accuracy and F-measure to compare our performance with other existing methods. Table 4.3 and Table 4.4 shows the comparison of accuracy and F-measure for Chinese Dataset and Dutch Dataset respectively. Comparison shows that Geometric feature method perform very poor in regards of accuracy. As we know that most of the geometric features are global features which is very prone to change. A small change in either rotation or scale geometric features vary significantly. That’s why it can not cope with intra-class variability. Moreover, these features are prone to noise which also affects the performance. Here our proposed method slightly outperforms other existing works. For other graph matching methods used earlier in signature verification, our method gains

Table 4.2: List of methods used for comparative analysis

Proposed by	Class	Year	Feature Descriptors	Training Model
Verma et al. [9]	WD	2013	Geometric Feature	Thresholding
Abuhaiba et al. [20]	WD	2007	Pixel Intensity	Graph Matching (hungarian)
Ghosh et al. [4]	WI	2014	Weighted Bipartite graph	Thresholding
Zouari et al. [14]	WD	2014	Fractal Dimension	K-nearest neighbor
Yilmaz et al. [1]	WD	2016	Local Binary Pattern (LBP)	Support Vector Machine
Proposed Method	WD	2018	Edge and node affinity Matrix	Factorized Graph Matching

a significant upgrade in accuracy. For a better visibility a graphical representation of this comparison is presented in Figure 4.6.

Table 4.3: Performance Comparison for Chinese Dataset

ID	Method	Accuracy	F - measure
1	Verma et al.	0.7150	0.7183
2	Abuhaiba et al.	0.7983	0.8000
3	Ghosh et al.	0.7700	0.7685
4	Zouari et al.	0.8317	0.8347
5	Yilmaz et al.	0.8317	0.8325
6	Proposed Method	0.8617	0.8591

Table 4.4: Performance Comparison for Dutch Dataset

ID	Method	Accuracy	F - measure
1	Verma et al.	0.7411	0.7437
2	Abuhaiba et al.	0.8042	0.8076
3	Ghosh et al.	0.7729	0.7746
4	Zouari et al.	0.8313	0.8307
5	Yilmaz et al.	0.8406	0.8413
6	Proposed Method	0.8500	0.8489

One of the most important application of offline signature verification is banking. As there is financial transaction, authentication accuracy is a bigger concern. More importantly, false positive rate should be as less as possible. The method with lower False Acceptance Rate (FAR) considered to be a better one for banking applications. Figure 4.7 illustrates the comparison of performance against False Acceptance Rate (FAR) and False Rejection Rate (FRR) for both the Chinese Dataset and the Dutch Dataset.

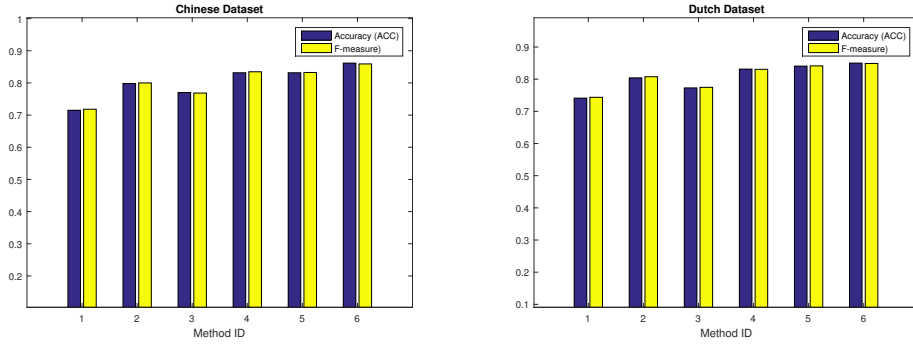


Figure 4.6: Comparison of Accuracy and F-measure

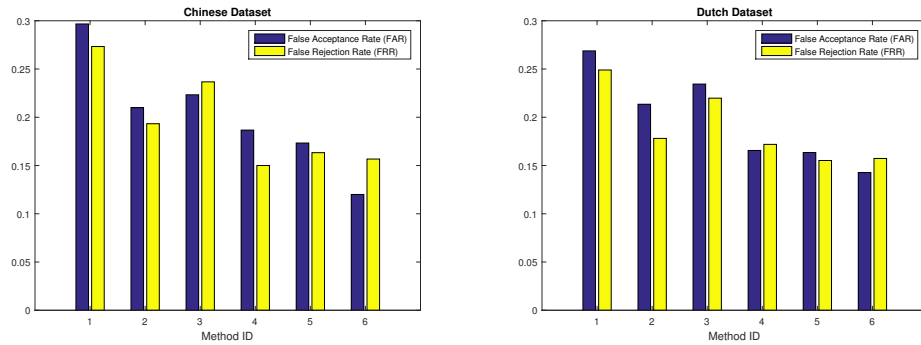


Figure 4.7: Comparison of False Acceptance Rate (FAR) and False Rejection Rate (FRR)

### 4.3.2 Comparison Regarding Computation Time

In real life applications like banking and other real time processes need faster operation. Requirements for offline signature verification is faster operation with better accuracy. To compare our method with existing works we kept time log for each method while running them in our system. Less amount of time indicates better performance regarding speed. Table 4.5 shows the speed comparison result.

Table 4.5: Comparison based on Computation Time

ID	Method	Time (ms)
1	Verma et al.	35.92
2	Abuhaiba et al.	95.23
3	Ghosh et al.	72.35
4	Zouari et al.	125.36
5	Yilmaz et al.	133.50
6	Proposed Method	<b>112.98</b>

Comparison here shows that methods with geometric feature performs better in terms of speed, but lacks accuracy. Our proposed method has a moderate speed

with good accuracy and lower FAR and FRR to make it more suitable for real life applications.

# Chapter 5

## Conclusion

### 5.1 Summary of the Contributions

Within the field of human identification, the usage of biometrics is growing because of its unique properties. The verifications are necessary for many routine activities such as boarding an aircraft, crossing international borders and entering a secure physical location. The higher levels of security and easier interactions to the end user are provided by biometrics for identity verification. Handwritten signature is the most popular and widely used biometric method. It is a necessity to have a robust, time efficient method to verify a signature. In this research we proposed an offline signature verification method to overcome the challenges and compete with other promising methods. Our method uses graph representation of a signature. We introduced an efficient way to compute the features which provide robustness and gives better performance with graph matching. As we know graph matching is time consuming and computationally costly. We proposed to use the efficient graph matching which uses factorization to reduce the computation. Comparison of performance shows that our system outperforms most of the state-of-the-art methods.

### 5.2 Future Works

While state-of-the-art in offline signature verification achieves around 12-25% FAR in various databases, the performance of these systems would be expected to be significantly worse with signatures collected in real life scenarios. In the future, systems research needs to concentrate on increasing the robustness of systems towards larger variations encountered in real life (e.g. signatures signed in smaller spaces, or in a hurry, or on documents with interfering lines).

Another issue is to allow the system work well with less number of references, such as three as is the case in many banking operations or even with one reference. We shall try to explore better feature representations of signature image (in particular learning representations from signature images with Deep Learning methods), and ways to improve classification with limited number of samples. In the paper we have illustrated the advantages of factorizing the pair-wise affinity matrix of typical graph

matching problems. The most computationally consuming part of the algorithm is the large number of iterations needed for FW method to converge when  $J_\alpha$  is close to a convex function. Therefore, more advanced techniques (e.g., conjugate gradient) can be used to speedup FW.

# Bibliography

- [1] M. B. Yilmaz and B. Yanıkoğlu, “Score level fusion of classifiers in off-line signature verification,” *Information Fusion*, vol. 32, pp. 109–119, 2016.
- [2] J. Ruiz-del Solar, C. Devia, P. Loncomilla, and F. Concha, “Offline signature verification using local interest points and descriptors,” in *Iberoamerican Congress on Pattern Recognition*, pp. 22–29, Springer, 2008.
- [3] M. I. Malik, M. Liwicki, A. Dengel, S. Uchida, and V. Frinken, “Automatic signature stability analysis and verification using local features,” in *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pp. 621–626, IEEE, 2014.
- [4] B. R. Ghosh, S. Banerjee, S. Dey, S. Ganguli, and S. Sarkar, “Off-line signature verification system using weighted complete bipartite graph,” in *Business and Information Management (ICBIM), 2014 2nd International Conference on*, pp. 109–113, IEEE, 2014.
- [5] C. Y. Suen, Q. Xu, and L. Lam, “Automatic recognition of handwritten data on cheques—fact or fiction?,” *Pattern Recognition Letters*, vol. 20, no. 11-13, pp. 1287–1295, 1999.
- [6] H. Baltzakis and N. Papamarkos, “A new signature verification technique based on a two-stage neural network classifier,” *Engineering applications of Artificial intelligence*, vol. 14, no. 1, pp. 95–103, 2001.
- [7] A. El-Yacoubi, E. Justino, R. Sabourin, and F. Bortolozzi, “Off-line signature verification using hmms and cross-validation,” in *NEURAL NETWORKS SIGNAL PROCESS PROC IEEE*, vol. 2, pp. 859–868, 2000.
- [8] E. J. Justino, A. El Yacoubi, F. Bortolozzi, and R. Sabourin, “An off-line signature verification system using hmm and graphometric features,” in *Proc. of the 4th international workshop on document analysis systems*, pp. 211–222, 2000.
- [9] A. C. Verma, D. Saha, and H. Saikia, “Forgery detection in offline handwritten signature using global and geometric features,” *IJCER*, vol. 2, no. 2, pp. 182–188, 2013.

- [10] W. F. Nemcek and W. C. Lin, "Experimental investigation of automatic signature verification," *IEEE Transactions on Systems, Man, and Cybernetics*, no. 1, pp. 121–126, 1974.
- [11] M. R. Pourshahabi, M. H. Sigari, and H. R. Pourreza, "Offline handwritten signature identification and verification using contourlet transform," in *2009 International Conference of Soft Computing and Pattern Recognition*, pp. 670–673, IEEE, 2009.
- [12] J. Coetzer, *Off-line signature verification*. PhD thesis, Stellenbosch: University of Stellenbosch, 2005.
- [13] P. S. Deng, H.-Y. M. Liao, C. W. Ho, and H.-R. Tyan, "Wavelet-based off-line handwritten signature verification," *Computer vision and image understanding*, vol. 76, no. 3, pp. 173–190, 1999.
- [14] R. Zouari, R. Mokni, and M. Kherallah, "Identification and verification system of offline handwritten signature using fractal approach," in *Image Processing, Applications and Systems Conference (IPAS), 2014 First International*, pp. 1–4, IEEE, 2014.
- [15] M. B. Yilmaz, B. Yanikoglu, C. Tirkaz, and A. Kholmatov, "Offline signature verification using classifier combination of hog and lbp features," in *Biometrics (IJCB), 2011 international joint conference on*, pp. 1–7, IEEE, 2011.
- [16] Y. Serdouk, H. Nemmour, and Y. Chibani, "Combination of oc-lbp and longest run features for off-line signature verification," in *Signal-Image Technology and Internet-Based Systems (SITIS), 2014 Tenth International Conference on*, pp. 84–88, IEEE, 2014.
- [17] Y. Serdouk, H. Nemmour, and Y. Chibani, "Orthogonal combination and rotation invariant of local binary patterns for off-line handwritten signature verification,"
- [18] J. Hu and Y. Chen, "Offline signature verification using real adaboost classifier combination of pseudo-dynamic features," in *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pp. 1345–1349, IEEE, 2013.
- [19] J. Vargas, M. Ferrer, C. Travieso, and J. B. Alonso, "Off-line signature verification based on grey level information using texture features," *Pattern Recognition*, vol. 44, no. 2, pp. 375–385, 2011.
- [20] I. S. Abuhaiba, "Offline signature verification using graph matching," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 15, no. 1, pp. 89–104, 2007.



- [21] A. Ramachandra, K. Pavithra, K. Yashasvini, K. Raja, K. Venugopal, and L. Patnaik, "Cross-validation for graph matching based offline signature verification," in *India Conference, 2008. INDICON 2008. Annual IEEE*, vol. 1, pp. 17–22, IEEE, 2008.
- [22] A. Ramachandra, J. Ravi, K. Raja, K. Venugopal, and L. Patnaik, "Signature verification using graph matching and cross-validation principle," *International Journal of Recent Trends in Engineering*, vol. 1, no. 1, p. 57, 2009.
- [23] "Comparison of the opencv's feature detection algorithms." [Online; accessed 11-April-2018].
- [24] E. Bayraktar and P. Boyraz, "Analysis of feature detector and descriptor combinations with a localization experiment for various performance metrics," 2017.
- [25] Ş. Işık, "A comparative evaluation of well-known feature detectors and descriptors," vol. 3, pp. 1–6, 2014.
- [26] E. Karami, S. Prasad, and M. Shehata, "Image matching using sift, surf, brief and orb: performance comparison for distorted images," 2017.
- [27] D. Rivard, E. Granger, and R. Sabourin, "Multi-feature extraction and selection in writer-independent off-line signature verification," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 16, no. 1, pp. 83–103, 2013.
- [28] G. S. Eskander, R. Sabourin, and E. Granger, "Hybrid writer-independent–writer-dependent offline signature verification system," *IET biometrics*, vol. 2, no. 4, pp. 169–181, 2013.
- [29] L. S. Oliveira, E. Justino, C. Freitas, and R. Sabourin, "The graphology applied to signature verification," in *12th Conference of the International Graphonomics Society*, pp. 286–290, 2005.
- [30] L. Batista, E. Granger, and R. Sabourin, "Dynamic selection of generative–discriminative ensembles for off-line signature verification," *Pattern Recognition*, vol. 45, no. 4, pp. 1326–1340, 2012.
- [31] E. Özgündüz, T. Şentürk, and M. E. Karşlgil, "Off-line signature verification and recognition by support vector machine," in *Signal Processing Conference, 2005 13th European*, pp. 1–4, IEEE, 2005.
- [32] E. J. Justino, F. Bortolozzi, and R. Sabourin, "A comparison of svm and hmm classifiers in the off-line signature verification," *Pattern recognition letters*, vol. 26, no. 9, pp. 1377–1385, 2005.
- [33] D. Bertolini, L. S. Oliveira, E. Justino, and R. Sabourin, "Reducing forgeries in writer-independent off-line signature verification through ensemble of classifiers," *Pattern Recognition*, vol. 43, no. 1, pp. 387–396, 2010.

- [34] R. Kumar, J. Sharma, and B. Chanda, “Writer-independent off-line signature verification using surroundedness feature,” *Pattern recognition letters*, vol. 33, no. 3, pp. 301–308, 2012.
- [35] L. G. Hafemann, R. Sabourin, and L. S. Oliveira, “Offline handwritten signature verification—literature review,” in *Image Processing Theory, Tools and Applications (IPTA), 2017 Seventh International Conference on*, pp. 1–8, IEEE, 2017.
- [36] Y. Guerbai, Y. Chibani, and B. Hadjadji, “The effective use of the one-class svm classifier for handwritten signature verification based on writer-independent parameters,” *Pattern Recognition*, vol. 48, no. 1, pp. 103–113, 2015.
- [37] K. Huang and H. Yan, “Off-line signature verification based on geometric feature extraction and neural network classification,” *Pattern Recognition*, vol. 30, no. 1, pp. 9–17, 1997.
- [38] B. Shekar, R. Bharathi, J. Kittler, Y. V. Vizilter, and L. Mestestskiy, “Grid structured morphological pattern spectrum for off-line signature verification,” in *Biometrics (ICB), 2015 International Conference on*, pp. 430–435, IEEE, 2015.
- [39] A. Soleimani, B. N. Araabi, and K. Fouladi, “Deep multitask metric learning for offline signature verification,” *Pattern Recognition Letters*, vol. 80, pp. 84–90, 2016.
- [40] H. Rantzsch, H. Yang, and C. Meinel, “Signature embedding: Writer independent offline signature verification with deep metric learning,” in *International symposium on visual computing*, pp. 616–625, Springer, 2016.
- [41] E. Rosten, R. Porter, and T. Drummond, “Faster and better: A machine learning approach to corner detection,” vol. 32, pp. 105–119, IEEE, 2010.
- [42] S. M. Smith and J. M. Brady, “Susan: a new approach to low level image processing,” in *International journal of computer vision*, vol. 23, pp. 45–78, Springer, 1997.
- [43] J. Bresenham, “A linear algorithm for incremental digital display of circular arcs,” in *Communications of the ACM*, vol. 20, pp. 100–106, ACM, 1977.
- [44] E. Rosten, “Fast corner detection.” [Online; accessed 13-September-2018].
- [45] D.-T. Lee and B. J. Schachter, “Two algorithms for constructing a delaunay triangulation,” *International Journal of Computer & Information Sciences*, vol. 9, no. 3, pp. 219–242, 1980.
- [46] G. Levi and T. Hassner, “Latch: learned arrangements of three patch codes,” in *2016 IEEE winter conference on applications of computer vision (WACV)*, pp. 1–9, IEEE, 2016.

- [47] H. Bay, T. Tuytelaars, and L. Van Gool, “Surf: Speeded up robust features,” in *European conference on computer vision*, pp. 404–417, Springer, 2006.
- [48] M. Zaslavskiy, F. Bach, and J.-P. Vert, “A path following algorithm for the graph matching problem,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 12, pp. 2227–2242, 2009.
- [49] M. Cho, J. Lee, and K. M. Lee, “Reweighted random walks for graph matching,” in *European conference on Computer vision*, pp. 492–505, Springer, 2010.
- [50] S. Gold and A. Rangarajan, “A graduated assignment algorithm for graph matching,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 18, no. 4, pp. 377–388, 1996.
- [51] B. J. van Wyk and M. A. van Wyk, “A pocs-based graph matching algorithm,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 11, pp. 1526–1530, 2004.
- [52] R. Zass and A. Shashua, “Probabilistic graph and hypergraph matching,” 2008.
- [53] M. Leordeanu, M. Hebert, and R. Sukthankar, “An integer projected fixed point method for graph matching and map inference,” in *Advances in neural information processing systems*, pp. 1114–1122, 2009.
- [54] K.-M. Ng, “A continuation approach for solving nonlinear optimization problems with discrete variables,” *Doctor Dissertation, Department of Management Science and Engineering of Stanford University*, 2002.
- [55] E. L. Allgower and K. Georg, *Introduction to numerical continuation methods*, vol. 45. SIAM, 2003.
- [56] I. M. Bomze and G. Danninger, “A global optimization algorithm for concave quadratic programming problems,” *SIAM Journal on Optimization*, vol. 3, no. 4, pp. 826–842, 1993.
- [57] J. Maciel and J. P. Costeira, “A global solution to sparse correspondence problems,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 2, pp. 187–199, 2003.
- [58] M. Frank and P. Wolfe, “An algorithm for quadratic programming,” *Naval research logistics quarterly*, vol. 3, no. 1-2, pp. 95–110, 1956.
- [59] M. Fukushima, “A modified frank-wolfe algorithm for solving the traffic assignment problem,” *Transportation Research Part B: Methodological*, vol. 18, no. 2, pp. 169–177, 1984.
- [60] A. A. Rahman, G. Mostaeen, and M. H. Kabir, “A statistical approach for offline signature verification using local gradient features,” in *International Conference on Electrical, Computer & Telecommunication Engineering (ICECTE)*, pp. 1–4, IEEE, 2016.

- [61] M. Liwicki, M. I. Malik, C. E. Van Den Heuvel, X. Chen, C. Berger, R. Stoel, M. Blumenstein, and B. Found, “Signature verification competition for online and offline skilled forgeries (sigcomp2011),” in *International Conference on Document Analysis and Recognition (ICDAR)*, pp. 1480–1484, IEEE, 2011.
- [62] T. Fawcett, “An introduction to roc analysis,” in *Pattern recognition letters*, vol. 27, pp. 861–874, Elsevier, 2006.