OFFLINE HANDWRITTEN SIGNATURE VERIFICATION USING SIFT FEATURES

BY

KARANJA EVANSON MWANGI
2006/HD18/6803K
BSc(Computer Science and Maths)(JKUAT)
Email: *emkaranja@yahoo.com*
Phone: +254-721-764076.

A Project Report Submitted to School of Graduate Studies in Partial Fulfillment for the Award of Master of Science in Computer Science Degree of Makerere university

OPTION: Computer Science

December, 2008

DECLARATION

I, Karanja Evanson Mwangi do hereby declare that this project report is original and has not been published and / or submitted for any other degree award to any other university before.

Signed:.............................. Date:...................

KARANJA EVANSON MWANGI
BSc Maths and Computer Science (JKUAT)
Department of Computer Science
Faculty of Computing and Information Technology, Makerere University

APPROVAL

This Project Report has been submitted for examination with the approval of the following supervisor.

Signed:... Date:...............................

DR JOHN QUINN, Ph.D.
Department of Computer Science
Faculty of Computing and Information Technology, Makerere University

DEDICATION

To my grandparents

Your wisdom, Iam only beginning to understand.

To my beloved parents Mr And Mrs George Karanja

For your support and innumerable words of encouragement when the ardor to complete this task and many others in my life seemed to wane.

To my brothers John and Crispus

For reminding me inch by inch and everything's cinch. Thanks for your love and friendship over the years.

"I’ve always thrived on the encouragement of others." Patti Smith

ACKNOWLEDGEMENT

With exception, I would like to express my sincere gratitude to the Almighty God who is full of mercy and compassion for giving me strength and good health during the whole period of my study.

I wish to extend my sincere thanks to my supervisor Dr. John Quinn for his time and nice ideas that shaped this work. I acknowledge the entire staff of the Faculty of Computing and Information Technology, especially my lecturers in Masters class for guidance and knowledge given to me while pursuing the course, not forgetting my course mates for their remarkable social and academic support. It is memorable to me. Special thanks goes to Dr. Patrick .J. Ogao for his invaluable advice as a friend and a mentor.

Iam highly indebted to my family for material support during the course of my study.

I will not forget to thank many friends I met during my stay in Uganda, some of whom contributed to this study by providing constructive critictism and sample signatures. Though I might not be able to name all these wonderful people by name. I sincerely extend my thanks and appreciation.

Thanks goes to Wairagu, Margaret, George, Dickson and Moses for openness and availability to discuss diverse social and academic issues.

I appreciate the friendship and support I got from Alois, Agnes, kris, Gerald, Miriam, Muiruri and Justin during the last but hard days of the research.

Much technical help came from many online discussion forums I engaged in particulary Matlab usergroup (www.mathworks.com). My thanks goes to the contributors of this forum especially Walter Roberson, who took his time and clarified my queries with e-mail communication which were very fruitful.

A great many people had a hand in ensuring the success of this work. If there's anybody I've forgotten to acknowledge, I apologize unreservedly . May God bless you more.
Be blessed and thanks to you all.

Contents

List of Figures

List of Tables

LIST OF ABBREVIATIONS AND ACRONYMS

TP	True Positive
FP	False Positive
FN	False Negative
TN	True Negative
FAR	False Acceptance Rate
HMM	Hidden Markov Model
NN	Neural Networks
SIFT	Scale Invariant Features Transform
AER	Average Error Rate
EER	Equal Error Rate
FRR	False Rejection Rate
HSV	Handwritten Signature Verification
FA	False Acceptance
SVM	Support Vector Machine
DoG	Difference-of-Gaussian

ABSTRACT

In this research we evaluate the use of SIFT features in offline handwritten signature verification. For each known writer we take a sample of three genuine signatures and extract their SIFT descriptors. We calculate the intra-class Euclidean distances (measure of variability within the same author) among SIFT descriptors of this known signatures. The keypoints Euclidean distances, the image distances and the intra class thresholds are stored as templates. We evaluate use of various intra-class distance thresholds like the maximum, average, minimum and range. For each signature claimed to be of the known writers, we extract its SIFT descriptors and calculate the inter-class distances, that is the Euclidean distances between each of its SIFT descriptors and those of the known template and image distances between the test signature and members of the the genuine sample. The intra-class threshold is compared to the inter-class threshold for the claimed signature to be considered a forgery. A database of 90 signatures consisting of a training set and a test set is used. The training set is made up of 54 genuine signatures from 18 known writers each contributing a sample of 3 signatures. The testset consists of 36 signatures, 18 genuine signature and 18 forged signature. The specificity and sensitivity of the verifier is measured and compared with the results from the analysis of human expert.

Chapter 1

BACKGROUND

1.1 Introduction

Handwritten signatures are widely accepted as a means of document authentication, authorization and personal verification. For legality most documents like bank cheques, travel passports and academic certificates need to have authorized handwritten signatures. In modern society where fraud is rampant, there is the need for an automatic HSV(Handwritten signature verification) system to complement visual verification.

Automated signature verification is as important as other automatic identification systems, though they differ from other systems that rely on possession of keys e.t.c or knowledge of specific personal information like passwords. They rely on well learned gestures and still they are most socially and legally accepted form of personal identification [3, 4].

Biometrics can be classified into two types; physiological and behavioural. Physiological biometrics measure some physical features of the subject like fingerprints, iris, hand and finger geometry which are stable over time. Behavioural biometrics measures user actions like speaking, writting and walking which are affected by health, age and physiological factors [5, 6]. A signature is a behavioural biometric characterised by behavioural trait that a writer learns and acquires over a period of time and becomes his unique identity [7, 5].

HSV systems are suited for forgery detection as they are cheap and nonintrusive and provide a direct link between the writer's identity and the transaction [1]. The objective of signature verification systems is to differentiate between original and forged signature, which are related to intra-personal and inter-personal variability [8, 9]. Intra-personal variation is variation among the signatures of the same person and inter-personal is the variation between the originals and the forgeries [9, 7].

We make a distinction between signature recognition and signature verification. Verification decides whether a claim that a particular signature belong to a specific class (writer) is true or false whereas recognition decides to which of a certain number of classes(writers) a particular signature belongs [1, 10, 5].

Automatic HSV systems are classified into two: offline HSV and online HSV [11, 12]. The online signature is captured using a special pen called a stylus and digitizing tablet and analysis is based on dynamic characteristics like pressure, velocity, acceleration and capture time of each point on the signature trajectory. In offline systems the input is a static image that is scanned and used for analysis. Both offline and online systems are used to detect various types of forgeries.
Signature forgeries are classified as follows [8, 13, 1, 4, 12]:

(i) Random/simple or zero effort. The forger doesnt have the shape of the writer signature but comes up with a scribble of his own. He may derive this from the writers name. This forgery accounts for majority of forgery cases though its easy to detect with naked eyes.

(ii) Unskilled /casual forgery. The forger knows the writers signature shape and tries to imitate it without much practice.

(iii) Skilled forgeries . This is where the forger has unrestricted access to genuine signature model and comes up with a forged sample.

The skilled forgery category has been classified further into amateur and professional forgery. A professional forgery is done by a person with professional expertise in handwriting analysis and is able to come up with high quality forgery.

The amateur forgeries are subcategorized in the context of online verification into home-improved and over-the-shoulder forgeries. Home-improved is when the forger has a paper copy of the signature and has ample time to practice at home. The imitation is based on static features of the image.

Over-the shoulder forgeries are produced when immediately the forger has witnessed the writer make a genuine signature, the forger in this case has dynamic properties of signature and spatial image [1, 4].

The Figure 1.1 shows the classification of forgeries.

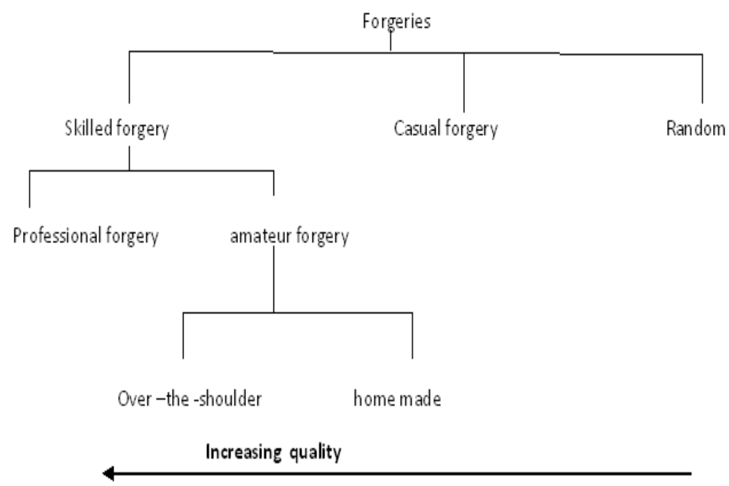


Figure 1.1: Forgery classification (reproduced from [1]).

1.2 Definition of Terms

Definition of some terms that are used in the project.

Definition 1: Pattern Matching is the science that concerns the description or classification of measurements based on underlying model [14].

Definition 2: False Rejection (FR) is when a genuine signature is rejected as a forged signature [13].

Definition 3: False Acceptance (FA) is when a forged signature is accepted as a genuine signature [13].

Definition 4: False Rejection Rate (FRR) is ratio of the number of genuine signatures rejected to the total number of genuine signatures submitted [1].

Definition 5: False Acceptance Rate (FAR) is ratio of the number of forged signatures accepted to the total number of forged signatures submitted [1].

Definition 6: Average Error Rate (AER) is the average of FAR and FRR [1].

Definition 7: Equal Error Rate (EER) is a point where FAR and FRR are equal [1, 4].

1.3 Statement of the Problem

Most of the available offline handwritten signature verification methods do not cater for scale and rotation variation. The Scale Invariant Feature Transform (SIFT) is an image processing algorithm that takes an image and transforms into a collection of local feature vector. Each of this feature vectors is distinctive and invariant to any scaling ,rotation or translation of the image . The SIFT algorithm has proved to be efficient in both recognition and verification problems but has not been used to solve HSV problems. Fraud, especially handwritten signature forgery, is rampant across all sectors of the economy, from universities to banks. Banks lose billions of dollars through fraudulent encashment of checks. According to *Ernst and Young* report, more than 500 million cheques are forged every year and this leads to more than $ 10 billion in losses [15] . The figure is predicted to grow at a rate of 2.5 % annually.

Here in Uganda, cheque fraud is rampant. The American embassy in Kampala has issued a business fraud warning [16]. The fraud scam involves criminals in Uganda who steal or intercept original checks drawn on American bank accounts, modify the information on the checks, and then use them to purchase goods from American vendors. It is estimated that $1 million was lost in 2006 due to this. Bank of Uganda reports that despite that cheque settlements constitutes the largest form of settlement in the banking sector, cheque fraud (which includes signature forgery) is the fastest growing financial crime in Uganda [17]. Training institutions lose credibility when incompetent persons impersonate to be their graduates in the labour markets with forged documents .

Despite this eminent problem of signature forgery there is no economical and reliable automated handwritten signature verification system available to be used across all sectors of the economy to suplement human verification.

1.4 Objectives of the Study

1.4.1 General Objectives of the Study

To offer an efficient and economically viable state of the art system for offline handwritten signature verification that can be used by various stakeholders .

1.4.2 Specific Objectives of the Study

The specific objectives of this research project were:

(i) Collection of handwritten signature samples to be used as training and testing set.

(ii) Algorithm design and implementation of signature feature extraction and pattern classification.

(iii Testing the accuracy of the verifier.

1.5 Scope

The project dealt with static images of handwritten signatures. The genuine signature samples taken from known writers and the forgery sets were generated imitating the genuine set. Only SIFT features were used as signature image descriptors.

1.6 Significance of the Study

The research project sought to evaluate use of SIFT in solving handwritten signature verification problems. SIFT descriptors are robust image descriptors and cheap to compute in terms of processing requirements compared with other methods like neural networks and they can easily be used in low resourced environments to reduce losses that arise from forged handwritten signatures and assist to make timely decision.

Chapter 2

LITERATURE REVIEW

Vigorous research has been pursued in handwriting analysis and pattern matching for a number of years. In the area of HSV, especially offline HSV, different technologies have been used and still the area is being explored. In this section we review some of the recent papers on offline HSV. The approaches used by different researchers differ in the type of features extracted, the training method, the classification and verification model used. The categorization for these approaches done here is influenced by classification used in [18].

2.1 Hidden Markov Model (HMM) based

The approach of Justino et al [19], uses the graphometric features, that is static features like the density of pixels and the pseudo dynamic features represented by axial slant. They employ grid segmentation and divide the signature image into four zones each with column containing cells with horizontal and vertical projections. Each column is converted to a characteristic vector assigned a numeric value. A HMM is used for the learning and verification process.

In [1], a system is introduced that uses only global features. A discrete radon transform which is a sinograph is calculated for each signature binary image at range of $0 - 360^\circ$, which is a function of total pixel in the image and the intensity per given pixel calculated using non overlapping beams per angle for X number of angles. Due to this periodicity, it is shift, rotation and scale invariant. A HMM is used to model each writer signature. The method achieves an AER of 18.4% for a set of 440 genuine signatures from 32 writers with 132 skilled forgeries.

2.2 Fuzzy Logic Based Approaches

In [20], global features of the signature like the skeleton of the pen trace and the structure of upper and lower envelope are used as shape descriptors. These are obtained by sampling upper and external points from the binary image of the signature. High pressure regions where the writer made more pressure or emphasis to is generated to a linear function that is be used for maximizing the correlation between the vertical and horizontal projections of the skeleton. For each of the above shape descriptors a multi- layer perception is assigned and the network is trained with a modified back propagation algorithm and the output of each individual network is combined through a fuzzy integral voter. Using a test set of 1000 signatures the approach obtained 90% true verification.

The authors in [21] propose the system that extracts angle features that are modelled in to a fuzzy model based on Takagi-Sugeno model. The model is extended to include structural parameters that account for variation in writers styles and changes in mood and the inputs are optimized to derive multiple rules. This approach obtained over 70% true verification.

2.3 Neural Networks

The proposed system in [22] uses structure features from the signatures contour, modified direction feature and additional features like surface area,length skew and centroid feature in which a signature is divided into two halves and for each half a position of the centre of gravity is calculated in reference to the horizontal axis. For classification and verification two approaches are compared the Resilient Backpropagation (RBP) neural network and Radial Basic Function(RBF) using a database of 2106 signatures containing 936 genuine and 1170 forgeries. These two classifiers register 91.21% and 88 % true verification respectively.

The approach of [5] attempts to combine online and offline HSV. For static images the scale, rotation and displacement invariance is represented as a normalized Fourier descriptor that is yielded through retracing the contour repeatedly and the results of the periodic function expressed as a Fourier series. For a dynamic image a speed function is used as a descriptor. The online retrace is compared with the offline image template. For classification a Multilayer Perception (MLP) neural network is used with one input layer, one hidden layer and one output layer. The results presented are for dynamic image.

2.4 Graph Matching

The work of Abuhaiba [4] avoids the use of features and uses only raw binary pixel intensities. Offline HSV problem is formulated as graph matching problem. A binary image is represented as graph with a set of vertices and edges, the goal is to get the minimum cost of matching which is represented as a classic form of assignment problem in graph theory. The method test 75 signatures for Skilled forgery and 300 signatures for random forgery. This reports 26.7% and 5.6% FAR, 26.7% and 5.6% EER for skilled and random forgeries respectively.

2.5 Statistical and Distance Classifiers

The uniqueness of writers' handwriting is mapped with that of the signature in Srihari et al[7]. The writer signs in a predefined space of 2×2 inches and rotation is normalized with the horizontal axis. The gradient, structural and concavity are used as image descriptors. The gradient detects the local features of the image and the concavity detects the relationship between the structural and the local features. The verification model is based on the bayesian classifier is that uses mean and variance measures to classify. The system uses two databases of signature with a total of 106 writers and 3960 samples and obtain FRR of 21.90% and 30.93% respectively .

The system used in [23] uses global descriptors and local features. The approach split the signature into regions(envelopes) and get the Centre of Gravity (CoG) of sub region and the distance made by the CoG and the strokes whitespaces. The learning algorithm used is C4.5 and the classifying method is based on a decision tree. The method uses 100 genuine signature and 300 forgeries from 20 people who consist of 15 Chinese and 5 people providing English signatures. For both cases over 90% success verification is reported.

A unique method is introduced in [14]. In this approach various features are extracted which include global features like image gradient, statistical features derived from distribution of pixels of a signature and geometric and topographical descriptors like local correspondence to trace of the signature. The classification involves obtaining variations between the signatures of the same writer and obtaining a distribution in distance space. For any questioned signature the method obtains a distribution which is compared with the available knowns and a probability of similarity is obtained using a statistical Kolmorogorv-Smirnov test. Using only 4 genuine samples for learning the method achieves 84% accuracy which can be improved to 89% when the genuine signature sample size is increased. *This method does not use the set of forgery signatures in the training/learning.*

The method in [8] uses the geometric centre for feature extraction. The centre is obtained through vertical and horizontal splitting of the image. The signatures used are taken at different time periods to show the intrapersonal variations. The classification is done through a Euclidean classifier model which is a measure of variance between any two image vectors. For testing 21 genuine signatures and 30 forgeries are used. A set of 9 signatures is used for training the model, FAR obtained are 2.08% , 9.75% and 16.36% for random ,simple and skilled forgeries repectively. The FRR for original signatures is 14.58%.

In [24], a system that adopts an expert examiner approach is used which employs a smoothness criterion. The basis is formed in that that skilled forgery signature greatly resemble genuine one at a global scale but they are less smooth. They derive a smoothness index as a ratio of non smooth segments to total extracted segments and combine it with global features like baseline shift, aspect ratio. Using a database of 1320 genuine signatures from 55 writers each contributing 24 signatures and 1320 skilled forgeries from 12 writers each imitating two signatures for each of the 55 initial writers an AER of 21.7% was achieved.

Fang et al [25] uses similar approach as [24] but uses crossing method and fractal dimension method to extract the smoothness feature which they combine with global features. A minimum distance classifier is used for verification. For a database of 55 writers, with 24 skilled forgeries and 24 genuine signatures for each writer. An AER of 17.3% was achieved.

The system introduced in Miike et al [26] uses displacement extraction approach, where the displacement function between any two pair of signatures is the sum of the squared Euclidean distance between them and a penalty that ensures the smoothness of the displacement function. Based on this displacement a measure of dissimilarity is obtained between the genuine and forged signature. A data base of 20 writers is used with 10 training signatures,10 signatures for genuine set and 10 for forgeries. An AER of 24.9% is achieved. The Euclidean distance is achieved when the mean vector and the variance are used for estimation.

Use of a set of contour features that can describe the internal and external feature of the signature is proposed in [27]. The verification is based on Mahalanobis distance classifier. The training and testing is done through leave-one-out method. A data base of 20 writers is used with 10 training signatures, 10 signatures for genuine set and 10 for forgeries. An AER of 11.4% is achieved [26]. Mahalanobis distance is achieved when a mean vector and the full covariance matrix of a given class is estimated and trained.

2.5.1 Limitations of Existing Statistical and Distance Classifiers

Most of the existing statistical and distance based classifiers deals with geometric and structural features of the signatures and they do not cater for scale, rotation ,transformation and affine variation.

2.6 Support Vector Machine

Support Vector Machines (SVMs) are machine learning algorithms that uses a high dimentional feature space and estimate differences between classes of given data to generalize unseen data. The system in [12] uses global, directional and grid features of the signature and SVM for classification and verification. The database of 1320 signatures is used from 70 writers. 40 writers are used for training with each signing 8 signatures thus a total of 320 signatures for training. For initial testing the approach uses 8 original signatures and 8 forgeries achieves FRR 2% and FAR 11%.

2.7 Incorporating a Prior Model

In [11], Lin et al infer that in practical cases a set of forgeries for testing is not available and propose a model like one used in [14] that only require the set of genuine signatures. They use a two stage approach with the training stage where learning parameter of the classifier is used and application stage with primary classifier to get the new user signature and final classifier to map the output of primary classifier and the mapping obtained at the training stage. It uses the global features that provide information about the whole structure of the signature. Grid gray features are obtained as a average gray value in each grid overlapped on the preprocessed image and pseudo-dynamic features descriptors like ink distribution. For each set of descriptors, the classifiers give the FRR and FAR for simple forgery as follows. Texture feature 25% and 30.56%; grid features 25.42% and 22.78%, global feature 42.08% and 27.22 % for FRR and FAR respectively.

2.8 SIFT Related Work

Proposed by David Lowe, Scale Invariant Features Transform (SIFT) is used to extract distinctive invariant features from images [2]. The SIFT algorithm is robust for identifying stable key locations in the scale- space of a grey scale image [2, 28]. It uses the following four steps to extract the set of descriptors from a given image [2].

(i) Scale-Space extrema detection.

(ii) Accurate Keypoint localisation.

(iii) Orientation assignment.

(iv) Keypoint description.

Step 1: Scale-Space extrema detection involves searching over all scales and location of the signature image to detect key points of all sizes. This is done using a difference-of-Gaussian (DoG) function to identify potential interest points that are invariant to scale and orientation [28].

For each octave of scale space, the image is convolved with Gaussian functions producing a set of scale space images. Adjacent Gaussian images are subtracted to produce difference-of -Gaussian images. After each octave the Gaussian image is halved and the process is repeated. Figure 2.1 illustrates the blurred images at different scales and the computation of difference -of- Gaussian (DoG).

The Scale-space of a signature image is defined as the function $L(x,y,\alpha)$,which is convolution of a variable scale Gaussian $G(x,y,\alpha)$ with an input signature image $I(x,y)$ as follows [2]:

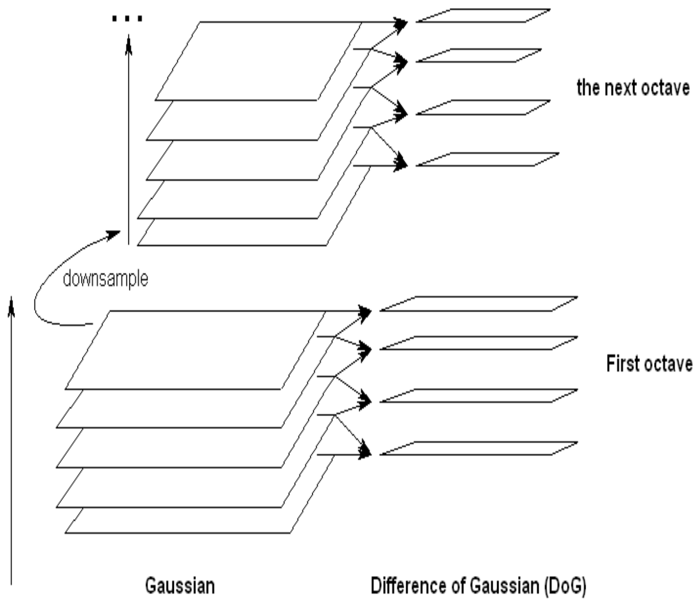


Figure 2.1: Difference -of- Gaussian computation.

$$L(x,y,\alpha) = G(x,y,\alpha) * I(x,y) \tag{2.1}$$

where * is the convolution in the *x* and *y* directions, and

$$G(x,y,\alpha)=\frac{1}{(2\pi\alpha^2)^{1/2}}exp(-\frac{x^2+y^2}{2\alpha^2}) \tag{2.2}$$

The difference between two nearby scales, $D(x,y,\alpha)$, separated by a constant multiplicative factor *k* is given by

$$D(x,y,\alpha) = (G(x,y,k\alpha) - G(x,y,\alpha)) * I(x,y) \tag{2.3}$$

$$= L(x,y,k\alpha) - L(x,y,\alpha) \tag{2.4}$$

The keypoints are identified as local maxima and minima of the DoG signature images across scale. Each pixel in the DoG is compared to other 8 neighbouring pixels at the same scale and 9 corresponding neighbours at the neighbouring scales. If the keypoint is the local maxima or minima, it is selected as a candidate keypoint. Figure 2.2 illustrates detecting the maxima and minima of difference-of-Gaussian in scale space.

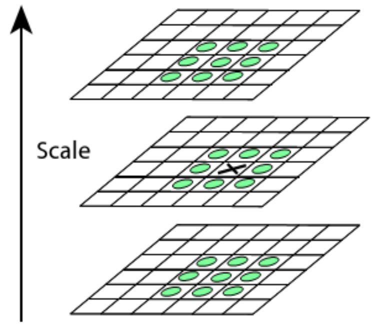


Figure 2.2: Scale space extrema detection (Reproduced from [2]).

Step 2: Accurate keypoint localisation. For each candidate keypoint identified, the interpolation of nearby data is used to accurately determine its point. Keypoints with low contrast (sensitive to noise) are dropped together with the responses poorly localised along the edges.
Step 3: Orientation Assignment. Each keypoint is assigned one or more orientations based on local image gradients directions. To determine the keypoint orientation, a gradient orientation histogram is computed in the neighborhood of the keypoint using the Gaussian image at the closest scale to the keypoints.

The contribution of each neighboring pixel is weighted by the gradient magnitude and a Gaussian window with α set to be 1.5 times the scale of the keypoint. This contributes to stability [2]. Peaks at the histogram are correspondent with dominant orientation. Any keypoint that is within 80% of the highest peak is used to create a separate keypoint. The orientation assignment of each keypoint is obtained by computing the gradient magnitude *M(x,y)* and orientation $\theta(x, y)$of the scale space for the scale of that keypoint:

$$M(x,y)=\sqrt{(K(x+1,y)-K(x-1,y))^2-(K(x,y+1)-K(x,y-1))^2} \tag{2.5}$$

and

$$\theta(x,y)=\arctan\frac{\mathrm{K(x,y+1)-K(x,y-1)}}{\mathrm{K(x+1,y)-K(x-1,y)}} \tag{2.6}$$

All the properties of the keypoint are measured relative to the keypoint orientation. This caters for rotation invariance.

Step 4: Keypoint Description. Local image gradients are measured at the selected scale in the region around each key point and transformed into a representation that allows local shape distortion and change in illumination.

When the keypoint orientation is selected, feature descriptors are computed as a set of orientation histograms on 4×4 pixel neighborhoods. The orientation histograms are relative to the keypoint orientation, and the orientation data comes from the Gaussian image closest in scale to the keypoints scale. The contribution of each pixel is weighted by the gradient magnitude and by a Gaussian with α 1.5 times the scale of the keypoint. Histograms contain 8 bins each and each descriptor contains an array of 4 histograms around the keypoint. This gives a SIFT feature with 4×4×8=128 values. This vector is normalized to enhance invariance to illumination.
SIFT features have the following advantages compared to other shape descriptors [2].

(i) Locality-Features detected are local and robust to clutter and occulsion.

(ii) Distinctiveness-Individual features can be matched to a large database.

(iii) Quantity -Many features can be generated even for small objects.

(iv) Efficiency for real time performance.

(v) Extensibility -They can be extended to different dimensions each with added robustness.

SIFT features have been used in pattern recognition and classification, mostly in object recognition. The work of Kim et al [29] uses SIFT features for robust digital watermarking. In [30] , the SIFT algorithm is used for face authentication using frontal view templates and evaluated for recognition of graffiti tags in [31] both with good results. Dlagnekov in his thesis used SIFT features for car make and model recognition with 89.5% true recognition rate [32]. More recently, use of SIFT features in fingerprint verification has been investigated [33]. Unlike these SIFT related work where the verification models have landmark features that have no intra class variability e.g. the location of the mouth and eyes in frontal view face authentication and minutiae points in fingerprint verification, which makes it easier to compute the nearest neighbours from these invariant points and do one to one mapping between the training class and the test class. Signatures have natural variance even among genuine signatures.

Chapter 3

METHODOLOGY

3.1 Introduction

Computer vision is often concerned with recognition of objects in a manner invariant to scale, pose, illumination and affine distortion. The SIFT algorithm takes an image and transforms it into a collection of local features where each of these feature vectors are distinctive and invariant to any scaling, rotation or translation of the image. In this project the SIFT features were considered. The implementation was done in MATLAB 6.0. The approach taken is a two step process with signature enrolment and verification. The forged signatures in the test set were generated by imitating the genuine signatures for each class on a piece of paper. The forgery was done by two people each generating a sample of three forged signatures per class which were given to a third party to chose one forgery which closely resembles the genuine set. Each forged signature was also scanned, cropped and stored in portable network graphic format. The results obtained from SIFT based verifier was compared with the results from human experts. Our original aim to use benchmark datasets from other research studies was not possible due to lack of cooperation and unavailability of online public datasets which are purely for offline handwritten signatures.

3.2 Steps Used in Offline Handwritten Signature Verification

The approach used for offline handwritten signature verification was broadly divided into two steps, signature enrolment and signature verification. Signature enrolment had four sub steps namely image pre-processing, extraction of SIFT features from signatures, calculation of Euclidean distances between images and creation of the known class signatures template. Signature verification had two sub steps namely outlier detection and comparison of test signature with known set so as to make a decision whether it is a genuine signature or not.

3.3 Signature Enrolment

Signature enrolment involved preparation of signatures, extraction of SIFT features and registration of signatures images and their SIFT features in the system.

3.3.1 Image Pre-Processing

The images used were signatures and were extracted from documents through scanning and cropping. A random sample of 18 signers was used, each signer contributed a sample of 3 signatures giving a total of 54 genuine signatures for the training set. The test set consisted of 18 genuine signatures and 18 forged signatures giving a total of 36 signatures for the test set. A database of 90 signatures was used in overall i.e. the training set and test set. Signature images were stored in portable network graphic (PNG) format. These images were converted to greyscale for further processing.

3.3.2 Extraction of SIFT Features From Signatures

This involved identifying stable shape descriptors from the pre processed signature image as described in Section 2.8 . The implementation that was used for extracting SIFT features was adopted from a MATLAB function written by El-Maraghi [34]. Figure 3.1 shows an example of scale space Gaussian images for one of the signatures in the test set. Figure 3.2 shows a sample signature and its keypoints and their orientation.

Figure 3.1: Example of space scale Gaussian images.

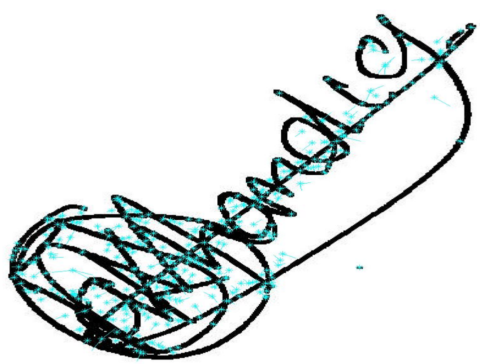


Figure 3.2: Example of a signature with extracted SIFT features.

3.3.3 Calculation of Euclidean Distances

This involved calculation of the Euclidean distances between the SIFT features of two given signature images to measure the variability between them. The motivation to use Euclidean distance as a measure of variability between images is derived from its success in object recognition [28] and lately in fingerprint verification [33]. Say we have two signatures **A** and **B**. Let A_i be the i^{th} keypoint in signature **A** and B_j be the j^{th} keypoint in signature **B**. The distance D(A_i,B_j) was calculated as the Euclidean distance between A_i and B_j. K_a, K_b are the number of keypoints in signature A and B respectively. The distance measure D(A_i,B) was taken as the average Euclidean distance from the ith keypoint in signature **A** to all the keypoints of signature **B**. The image distance between signature **A** and signature **B** is given by :

$$D(A,B) = \frac{1}{K_a} \sum_{i=1}^{K_a} D(A_i, B) \tag{3.1}$$

3.3.4 Creation of the Known Signature Template.

The implementation focused on upholding anonymity of the signers. Only the signatures and arbitrary writer IDs were used. For each known writer, a sample of three signatures say **A**, **B** and **C** were taken to cater for intra-personal variations. A template was generated as a MATLAB file and stored. The template has the following:

(i) Writer ID.

(ii) The Euclidean distances between keypoints i.e. D(A_i,B), D(A_i,C), and D(B_j,C).

(iii) The distances between the Signature images i.e. D(A,B), D(A,C) and D(B,C).

(iv) Intra-class thresholds: The maximum among $D(A,B), D(A,C)$ and $D(B,C)$ i.e. $max\,(D(A,B), D(A,C), D(B,C))$. The minimum among $D(A,B), D(A,C)$ and $D(B,C)$ i.e. $min\,(D(A,B), D(A,C), D(B,C))$. The average on $D(A,B), D(A,C)$ and $D(B,C)$ i.e. $avg\,(D(A,B), D(A,C), D(B,C))$. The range on maximum intra-class distance given by $max\,(D(A,B), D(A,C), D(B,C)) \pm 0.05$. The range on minimum intra -class distance given by $min\,(D(A,B), D(A,C), D(B,C)) \pm 0.05$.

Figure 3.3 is an example of a sample of three genuine signatures of a known writer taken to cater for intra-personal variation.

Figure 3.3: Example of intra-personal variation.

Figure 3.4 Summarizes the signature enrolment stage.

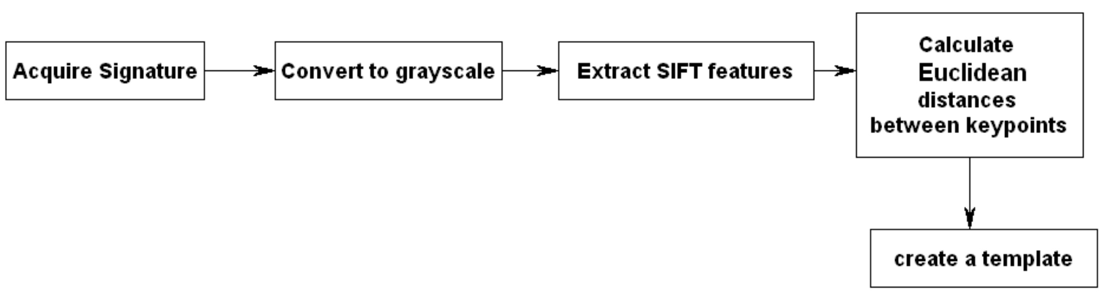


Figure 3.4: Steps in signature enrolment.

3.4 Signature Verification

Verification is the process of testing whether a claimed signature is of the same (class) writer as the set of signatures enrolled in the system for that class. Verification involved loading the template MATLAB file enrolled in the system and comparing its stored parameters with those calculated by the outlier detection process.

3.4.1 Outlier Detection

Given a test signature say T claimed to be of a particular writer, the Euclidean distances were calculated between the test signature and each of the three sample signatures (as discussed in Subsection 3.3.3) resulting to distances between the images i.e. D(T,A), D(T,B) and D(T,C).
The inter-class thresholds, $max\,(D(T,A), D(T,B), D(T,C))$, $min\,(D(T,A), D(T,B), D(T,C))$, $avg\,(D(T,A), D(T,B), D(T,C))$ are computed.

3.4.2 Comparison and Decision Criteria

The comparison between the distance parameters of the SIFT features of the claimed test signature was done with those of the stored template. Each decision criteria was a binary classification and was taken independently. We let W be $(D(T,A), D(T,B), D(T,C))$ and Z be $(D(A,B), D(A,C), D(B,C))$.

Test 1: Comparing inter-class maximum distance with intra-class maximum distance as threshold.
We classify T as genuine if the condition

$$max\,(Z) > max\,(W) \tag{3.2}$$

holds, otherwise we classify T as not genuine.

Test 2: Comparing average of inter-class distances with the average of intra-class distance as threshold.
We classify T as genuine if the condition

$$avg\,(Z) > avg\,(W) \tag{3.3}$$

holds, otherwise we classify T as not genuine.

Test 3: Comparing inter-class minimum distance with intra-class minimum distance as threshold.
We classify T as genuine if the condition

$$min\,(Z) > min\,(W) \tag{3.4}$$

holds, otherwise we classify T as not genuine.

Test 4: Using a range of 0.05 on the maximum intra-class distance as a threshold and comparing with inter-class maximum distance.

We classify T as genuine if the condition

$$max\,(Z) \pm 0.05 > max\,(W) \tag{3.5}$$

holds, otherwise we classify T as not genuine.

Test 5: Using a range of 0.05 on the minimum intra-class distance as a threshold and comparing with inter-class minimum distance.

We classify T as genuine if the condition

$$min\,(Z)) \pm 0.05 > min\,(W)) \tag{3.6}$$

holds, otherwise we classify T as not genuine.

Test 6: Using a range of 0.05 on both the minimum intra-class distance and minimum intra-class distance as a threshold such that the minimum and maximum inter- class distance should lie within that range.

We classify T as genuine if the condition

$$max\,(Z) \pm 0.05 > max\,(W)\ \ and\, min\,(Z) \pm 0.05 > min\,(W) \tag{3.7}$$

holds, otherwise we classify T as not genuine.

Figure 3.5 summarizes the signature enrolment and verification .

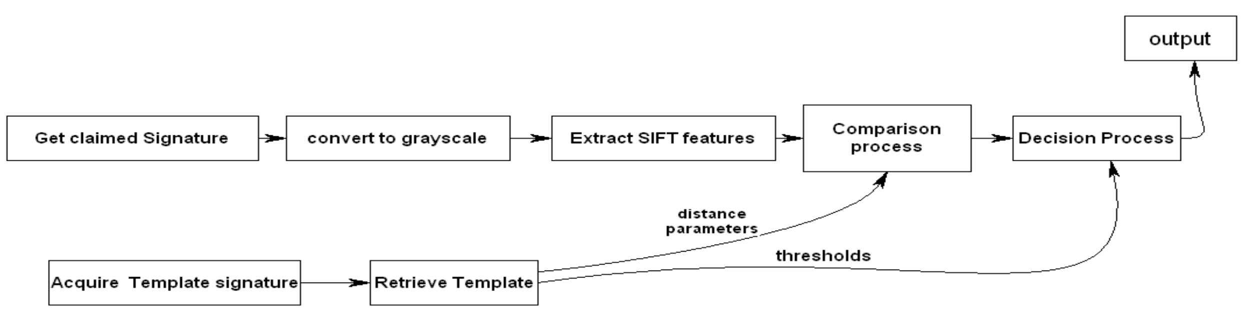


Figure 3.5: Flowchart showing signature enrolment and verification.

3.5 Measurement of the Signature Verifier Accuracy

To measure the accuracy of the verifier, a set consisting of genuine signatures and forged signatures was used and various performance statistics were used. These statistics are standard in machine learning literature, see example in Section 5.7 of [35].

(i) **True Positive** (TP) - A classification is a true positive if the signature is genuine (of known writer) and the output of the verifier ascertains that.

(ii) **False Positive** (FP) - A classification is a false positive if the signature is forged and the output of the verifier claims that it is genuine.

(iii) **True Negative** (TN) - A classification is a true negative if the signature is forged and the output of the verifier ascertains that.

(iv) **False Negative** (FN) - A classification is a false negative if the signature is genuine (of known writer) and the output of the verifier claims that it is forged.

(v) **The sensitivity** is the proportion of actual positives (genuine signatures) which are correctly identified as positives. which is given by:

$$Sensitivity = \frac{TP}{TP + FN} \tag{3.8}$$

(vi) **The specificity** is the proportion of negatives (forgeries) which are correctly identified, which is given by:

$$Specificity = \frac{TN}{TN + FP} \tag{3.9}$$

The test for accuracy of the system is summarised in Figure 3.6 :

		Actual Condition(Truth)	
		(+ve)Genuine	(-ve)Forgery
OUTPUT OF THE SYSTEM	(+ve)Genuine	**TP**	**FP**
	(-ve)Forgery	**FN**	**TN**

Figure 3.6: Confusion matrix for analysing accuracy.

3.6 Comparison with Human Expert

Four human experts were used; two bankers, a loan officer and a forensic accountant. These experts have wide experience in different working environments in the financial and business sector where signature forgery is rampant. Among their daily routines is to verify signatures before transactions are authorised.

The human experts were given the same sample of three signatures for each class of known writer to study their features . This training set and test set were the same ones earlier used as training set and test set in SIFT based verification. For each class, a test set of two signatures was given containing one forged signature and one genunine signature. Each of the test signature was compared with its class of knowns independently.

For each class they studied the features of the three known signatures and based on those features classify each the two test signatures as either genuine or forgery. To measure the accuracy of a human expert the same performance statistics used in Section 3.4.1 were computed and the results compared with those of the SIFT based classifier.

3.7 The Proposed Algorithm

The algorithm used can be summarised as follows:

(i) Given the set of known signatures and test signatures signed in a document, scan and crop each class of knowns and its respective test signatures and save them as portable network graphic (PNG) format.

(ii) For each signature in the class of known signatures say **A, B, C** and test signature **T**, perform SIFT extraction as described in Subsection 3.3.2.

(iii) For each pair of known signatures **A,B**, Let A_i be the i^{th} keypoint in signature **A** and B_j be the j^{th} keypoint in signature **B**. Calculate Euclidean distance D(A_i,B_j) and the distance D(A_i,B), the average distance from the i^{th} keypoint in signature **A** to all keypoints of signature **B**

(iv) Calculate image distance $D(A, B)$ as shown in Equation 3.1.

(v) Create the template of known signatures class consisting of writer ID, distance parameters and intra - class thresholds.

(vi) For a given test signature **T** claimed to be of a known writer, Calculate the inter- class distances between **T** and each signature in the class of knowns in the template. Get the inter-class thresholds.

(vii) Compare the intra - class thresholds in the template with inter- class thresholds using conditions set in Subsection 3.4.1.

(viii) Test the performance of the classifier using the performance statistics described in Section 3.5.

Chapter 4

RESULTS

4.1 Introduction

To measure the accuracy of the SIFT based verifier, a set consisting of genuine signatures and forged signatures was used. In total 90 signatures were used. The training set had 54 genuine signatures for creating the known signature templates. A test set consisted of a total of 36 signatures (18 genuine signatures and 18 forged signatures). For each class of known signatures containing three sample signatures, a genuine and a forged signature were tested independently. The overall performance of the SIFT based classifier was measured in terms of the number of genuine and forged signatures it can correctly classify in the test set.

4.2 Examples of Verified Signatures

In this Section we present examples of verified signatures. Figure 4.1 shows signatures 16.png, 17.png and 18.png from the same known writer(same class) and were used as the training set for this class to create a template. The signatures 19.png in Figure 4.2 was the test signature. Using all the five tests described in Subsection 3.4.2, signature 19.png was correctly identified as genuine. Table 4.1 shows the image distances between the set of known signatures 16.png, 17.png and 18.png. The intra class maximum, $max\,(D(16,17), D(17,18), D(16,18)) = 1.1710$ is greater than the inter class maximum $max\,(D(16,19), D(17,19), D(18,19)) = 1.0700$. The intra class average, $avg\,(D(16,17), D(17,18), D(16,18)) = 1.1293$ is greater than the inter class average

$avg\,(D(16,19), D(17,19), D(18,19)) = 1.0497$, the intra class range on maximumm intra class distances is 1.2210 is also greater than inter class maximum $max\,(D(16,19), D(17,19), D(18,19)) = 1.0700$. The intra-class minimum $min\,(D(16,17), D(17,18), D(16,18)) = 1.1069$ is greater than inter class minimum distance which is 1.0382.
Also the range on minimum, $min\,(D(16,17), D(17,18), D(16,18))$- 0.05$= 1.0569$ is also greater than inter class minimum. Hence based on all the tests signature 19.png is correctly classified as genuine. Table 4.2 shows the inter- class distances between the test signature 19.png and the template of knowns.

Figure 4.1: Example 1 of genuine signatures of a known writer.

Figure 4.2: Test signature correctly classfied as genuine by all the tests.

Table 4.1: Image distances set of known signatures 16.png, 17.png and 18.png.

Signatures	**Distance description**	**Image distance**
16.png,18.png	D(16,18)	1.1069
17.png,18.png	D(17,18)	1.1710
16.png,17.png	D(16,17)	1.1099

Table 4.2: Image distances between test signature 19.png and set of known signatures.

Signatures	**Distance description**	**Image distance**
16.png,19.png	D(16,19)	1.0411
17.png,19.png	D(17,19)	1.0700
18.png,19.png	D(18,19)	1.0382

Figure 4.3 shows signatures 41.png, 42.png and 43.png from the same known writer and were used as the training set for this class to create a template. Using this template, signature 45.png shown in Figure 4.4 was correctly classified as a forgery by all the tests. Table 4.3 shows the intra - class distances between signatures 41.png, 42.png and 43.png. Table 4.4 shows the inter - class distances between known signatures 41.png, 42.png, 43.png and test signature 45.png.

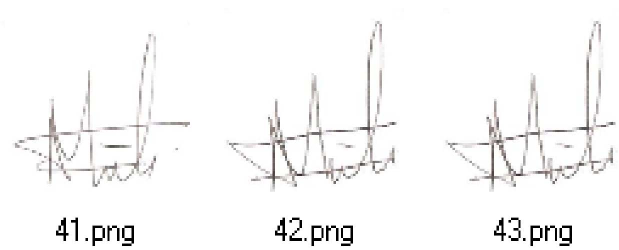
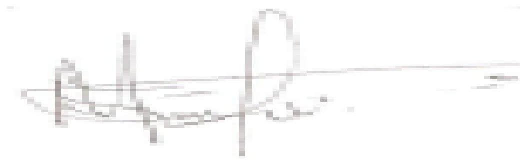


Figure 4.3: Example 2 of genuine signatures of a known writer.



45.png

Figure 4.4: Test signature correctly classfied as forgery by all the tests.

Table 4.3: Image distances set of known signatures 41.png, 42.png and 43.png.

Signatures	Distance description	Image distance
41.png,42.png	D(41,42)	1.0538
41.png,43.png	D(41,43)	1.0538
42.png,43.png	D(42,43)	1.1028

Table 4.4: Image distances between test signature 45.png and set of knowns 41.png, 42.png and 43.png.

Signatures	Distance description	Image distance
41.png,45.png	D(41,45)	1.2012
42.png,45.png	D(42,45)	1.3967
43.png,45.png	D(43,45)	1.0539

4.3 Results from the Proposed Method

MATLAB scripts were used to detect false positives, true positives, true negatives, true positives and to calculate the sensitivity and the specificity. Sensitivity is proportion of genuine signatures the classifier is able to correctly identify as genuine from the test set and the specificity is the proportion of the forgeries the classifier is able to correctly classify as forgeries from the test set. The following statistics were obtained.

4.3.1 Maximum Distance

The specificity of 38.89% was obtained; which is the proportion of forgeries the classifier was able to identify from the testing set and the sensitivity of 77.78% was also obtained; which is the proportion of genuine signatures the classifier was able to correctly identify after using the condition set in Equation 3.2, that is comparing the maximum intra-class distance with maximum inter-class distance. This means the comparison between the maximum intra - class distance and maximum inter - class distance was better in identifying genuine signatures than in detecting forgeries. Table 4.5 shows the performance statistics obtained by the classifier using maximum class distances.

Table 4.5: Performance statistics obtained by the classifier using maximum class distances.

TP	14	**FP**	11
TN	7	**FN**	4

4.3.2 Average Distance

Using the condition set in Equation 3.3, that is comparing the average intra-class distance with average inter-class distance. The specificity of 50% was obtained, which is the proportion of forged signatures correctly identified from the test set and the sensitivity of 44.444% was also obtained, that is the proportion of genuine signatures correctly identified. From these performance statistics it shows the average test was poor and random in both detecting the forged signatures and identifying the genuine signatures. Table 4.6 shows the performance statistics obtained by the classifier using average class distances.

Table 4.6: Performance statistics obtained by the classifier using average class distances.

TP	8	**FP**	9
TN	9	**FN**	10

4.3.3 Minimum Distance

The specificity of 38.889% and the sensitivity of 44.444% were obtained after using the condition set in Equation 3.4, that is comparing the minimum intra-class distance with minimum inter-class distance. Similar to the average test, the minimum distance test performed poorly in both detecting the forged signatures and identifying the genuine signatures. Table 4.7 shows the performance statistics obtained by the classifier using minimum class distances.

Table 4.7: Performance statistics obtained by the classifier using minimum class distances.

TP	7	**FP**	10
TN	8	**FN**	11

4.3.4 Range of ± 0.05 on Maximum Distance

The specificity of 33.3% and the sensitivity of 88.8% were obtained after using the condition set in Equation 3.5, that is a range of 0.05 on the maximum intra-class distance and setting it as a threshold and comparing it with the maximum inter-class distance. This test was the best in terms of sensitivity i.e. was able to correctly classify highest number of genuine signatures from the test set and the poorest in terms of specificity i.e. identifying forged signatures. Table 4.8 shows the performance statistics obtained by the classifier using the range test on maximum intra class distance.

Table 4.8: Performance statistics obtained by the classifier using the range test on maximum class distances.

TP	16	**FP**	15
TN	3	**FN**	2

4.3.5 Range of ± 0.05 on Minimum Distance

The specificity of 72.2% and the sensitivity of 50% were obtained after using the condition set in Equation 3.6, that is a range of 0.05 on the minimum intra-class distance and setting it as a threshold and comparing it with the minimum inter-class distance. This test was the best in identifying the forged signatures from the test set. Table 4.9 shows the performance statistics obtained by the classifier using the range test on minimum intra class distance .

Table 4.9: Performance statistics obtained by the classifier using the range test on minimum class distances.

TP	9	**FP**	5
TN	13	**FN**	9

4.3.6 Range of ± 0.05 on Maximum Distance and Range of ± 0.05 on Minimum Distance

The specificity of 55.5% and the sensitivity of 77.78% were obtained after using the condition set in Equation 3.7, that is a a range of 0.05 on both the minimum and maximum intra-class distances and setting them as a threshold. Table 4.10 shows the performance statistics obtained by the classifier using the range on both minimum and maximum intra-class distances. A good classifier should have high rates of both specificity and sensitivity. It should be able to correctly classify high proportion of genuine signatures from the test set and also detect high proportion of forged signatures as forgeries in the same test set. From the performance statistics, this test compared to the rest had high rates on both specificity and sensitivity and was considered for comparison with human experts.

Table 4.10: Performance statistics obtained by the classifier using the range test on both minimum and maximum class distances.

TP	14	**FP**	8
TN	10	**FN**	4

4.4 Results from Human Experts

The first human expert is a loan officer and an accountant with Holistic Services Uganda (HOSU) which local Non-Governmental Organisation. The first human expert obtained a sensitivity of 56.566% and specificity of 61.1 %. The performance statistics obtained by the first expert are shown in Table 4.11.

Table 4.11: Performance statistics obtained by the first human expert.

TP	10	**FP**	7
TN	11	**FN**	8

The second human expert is a banker with bank of Baroda Uganda. The second human expert obtained a sensitivity of 72.22% and specificity of 77.7 %. Table 4.12 shows the performance statistics obtained by the second human expert.

Table 4.12: Performance statistics obtained by the second human expert.

TP	13	**FP**	4
TN	14	**FN**	5

The third human expert is a also a banker with Stanbic bank Uganda. The third human expert obtained a sensitivity of 66.6% and specificity of 72.2 %. Table 4.13 shows the performance statistics obtained by the third human expert.

Table 4.13: Performance statistics obtained by the third human expert.

TP	12	**FP**	5
TN	13	**FN**	6

The fourth expert is forensic accountant with VAS consultants Ltd, which is a regional management consultancy. The fourth human expert obtained a sensitivity of 94.4% and specificity of 77.7 %.Table 4.14 shows the performance statistics obtained by the fourth human expert.

Table 4.14: Performance statistics obtained by the fourth human expert.

TP	17	**FP**	4
TN	14	**FN**	1

On average the human experts obtained a sensitivity of 72.445% and specificity of 72.175 %. Table 4.15 shows the summary of performance statistics obtained by the human experts.

Table 4.15: Summary of performance statistics obtained by the human experts.

Human Experts	**1**	**2**	**3**	**4**	**Average**
Sensitivity	56.56	72.22	66.6	94.4	72.445
Specificity	61.1	77.1	72.2	77.7	72.175

4.5 Comparison of Human Experts and Proposed Algorithm

The SIFT based classifier performed better in identifying genuine signatures compared to the average of human experts and was out performed in identifying forgeries by the human experts. SIFT tests were poor on average in specificity. Table 4.16 shows the sensitivity and specificity obtained by the human experts and the best of the SIFT method. The variation in performance statistics among the human expert was attributed to the diffrence in their working environments in terms of the kind of clients they deal with and physiological factors.

Table 4.16: Comparison of sensitivity and specificity obtained by the human experts and the proposed SIFT method.

Perfomance Statistic	**Average of Human Experts**	**SIFT method**
Senstivity	72.445	77.78
Specificity	72.175	55.5

Chapter 5

CONCLUSIONS AND AREAS OF FURTHER RESEARCH

5.1 Conclusions

The objective of this project was mainly to offer an efficient and economically viable offline handwritten signature verifier. In order to meet the objective various existing methods of offline handwritten signature verification were reviewed and SIFT features were decided as robust image descriptors. A database of signatures was collected consisting of known writers' signatures and forgeries. The efficiency of the verifier was tested and specificity and the sensitivity were measured for each test taken. It was noted that some writers have large discrepancies between three of their sample signatures such that even a forgery may fall within the intra class distances which may result to a false negative notification this might have been caused by physiological factors. A good classifier should have high rates of specificity and sensitivity. To be able to have an efficient classifier we picked the test that had high rates of both specificity and sensitivity. The optimal condition was given by Equation 3.7 that is, using a range of 0.05 on both the minimum intra-class distance and minimum intra-class distance as a threshold such that the minimum and maximum inter- class distance should lie within that range. Though originally designed for object recognition, the use of SIFT features for signature verification had not been systematically investigated before. The performance stastistics obtained from this test showed that SIFT features can be used with Euclidean distances for offline handwritten verification. Although this research is a good start to SIFT based handwritten signature verification it can be extended to evaluate other image similarity measures.

5.2 Areas of Further Research

The problem of handwritten signature verification was addressed from an offline point of view in the experiments. Many areas of study related to SIFT features and various distance measures are still open.

5.2.1 Alternative Distance Measures

Use of SIFT features as signature descriptors and other distance measures could be interesting. Chernoff-Bhattacharya distance, has been successfully used to measure discrminability in handwritten numeral recognition [36] could be evaluated in HSV problems.

Mahalanobis distance is another measure that can be used to find patterns in SIFT features . Unlike the Euclidean distance that uses the mean vector, Mahalanobis distance uses both the mean vector and the full covariance matrix which can an efficient measure of variability among signatures. If the covariance matrix is the identity matrix, the Mahalanobis distance reduces to the Euclidean distance. Detailed explanations of the Chernoff-Bhattacharya distance and Mahalanobis distance can be found in Chapter 6 of [37]. The experiments can also be extended to combine two or more of these distance measures and compare their efficiency.

5.2.2 SIFT Features and Online Handwritten Signature Verification

Since online handwritten signature verification problems involves descriptors like velocity, acceleration and capture time of each point on the signature trajectory. Future work could evaluate inclusion of SIFT features as image descriptors and various distance measures discussed above in online handwritten signature verification problems.

Chapter 6

APPENDICES

6.1 Appendix A

Here we outline various MATLAB scripts and functions that were used in this project.

6.1.1 MATLAB Functions

+++-

This function reads a signature image from file and converts it to grayscale.

```
function S = imreadbw(file)
S=im2double(imread(file));
if(size(S,3) > 1)
  S = rgb2gray( S );
end
```

++

This function resizes the displayed images .

```
function resizeImageFig(h, sz, frac)
if (nargin <3)
 frac = 1;
end
pos = get(h, ’Position’);
set(h, ’Units’, ’pixels’, ’Position’, ...
       [pos(1), pos(2)+pos(4)-frac*sz(1), ...
        frac*sz(2), frac*sz(1)]);
set(gca,’Position’, [0 0 1 1], ’Visible’, ’off’);
```

++

Calculates the intra-class Euclidean distances and the intra-class thresholds.

```
function[D13,D23,D12,AGD12,AGD13,AGD23,
intraMin,intraMax,intraAvg,maxRange,minRange]
= intraclassEuclidean( desc1,desc2,desc3)
for i = 1:size(desc1,1)
   D12 = sqrt(sum((desc2 - repmat(desc1(i,:),size(desc2,1),1)).^2,2));
    D13 = sqrt(sum((desc3 - repmat(desc1(i,:),size(desc3,1),1)).^2,2));
end
for i = 1:size(desc2,1)
    D23 = sqrt(sum((desc3 - repmat(desc2(i,:),size(desc3,1),1)).^2,2));
      AGD12=sum(D12)/size(desc2,1);
      AGD13=sum(D13)/size(desc3,1);
      AGD23=sum(D23)/size(desc3,1);
      d=[AGD12,AGD13,AGD23];
      intraMin=min(d);
      intraMax=max(d);
      intraAvg=sum(d)/3;
      maxRange=intraMax + 0.05;
```

```
      minRange=intraMin -0.05;

end
     end
```

++

calculates the inter-class Euclidean distances and the inter-class thresholds

This are the distances between the claimed signature and the template of the known writer.

```
function [D41,D42,D43,AGD41,AGD42,AGD43,interMin,interMax,interAvg]
=interclassEuclidean(desc1,desc2,desc3,desc4)
for i = 1:size(desc4,1)
   D41 = sqrt(sum((desc1 - repmat(desc4(i,:),size(desc1,1),1)).^2,2));
    D42 = sqrt(sum((desc2 - repmat(desc4(i,:),size(desc2,1),1)).^2,2));
      D43 = sqrt(sum((desc3 - repmat(desc4(i,:),size(desc3,1),1)).^2,2));
      AGD41=sum(D41)/size(desc4,1);
      AGD42=sum(D42)/size(desc4,1);
      AGD43=sum(D43)/size(desc4,1);
      d=[AGD41,AGD42,AGD43];
      interMin=min(d);
      interMax=max(d);
      interAvg=sum(d)/3;

end
```

++-

6.1.2 MATLAB Scripts

In this part we explore the scripts used in the project. The scripts have inline comments for ease of reference.

ENROLsignature.m

This script conducts the signature enrolment stage. It calls the functions that extracts the SIFT features of the three samples of the known writer signatures, calculate the Euclidean distances and the intra-class thresholds. The output is stored as a matlab file (template) which contain individual keypoint of each the signatures, the Eucledian distances between the individual keypoints, the distance between images and the intra-class thresholds.

```
clear;
close all;
global EvansSigMsc;
Signa_path = [EvansSigMsc ’E:\SIGNATURE_EVANS’];
addpath( Signa_path );
im_path = [Signa_path,’/Signatures/’];
keypoint_path = [Signa_path,’/KEYPOINTS/’];
octaves = 4;
intervals = 2;
cache = 1;
im_name1= input(’Please enter the ID of
known writer first signature \n’,’s’);
im1=im2double(imreadbw([im_path,im_name1,’.png’])) ;
im_name2= input(’Please enter the ID of
known writer second signature \n’,’s’);
im2=im2double(imreadbw([im_path,im_name2,’.png’])) ;
im_name3= input(’Please enter the ID of
known writer third signature \n’,’s’);
im3=im2double(imreadbw([im_path,im_name3,’.png’])) ;
fprintf( 2, ’Extracting SIFT descriptors(keypoints) for the
known writer  signatures.\n’ )
[pos1, scale1, orient1, desc1 ] = SIFT( im1, octaves, intervals,
ones(size(im1)), 0.02, 10.0, 1 );
```

```
[pos2, scale2, orient2, desc2] = SIFT( im2, octaves, intervals,
ones(size(im2)), 0.02, 10.0, 1 );
[pos3, scale3, orient3, desc3 ] = SIFT( im3, octaves, intervals,
ones(size(im3)), 0.02, 10.0, 1 );
fprintf( 2, 'Calculating Euclidean distances between keypoints
and intra-class threshold.\n' )
[[D13,D23,D12,AGD12,AGD13,AGD23,intraMin,
intraMax,intraAvg,maxRange,minRange]
 = intraclassEuclidean( desc1,desc2,desc3);
fprintf( 2, 'Saving a template for known writer signatures.\n' );
key_name1= ([im_name1,im_name2,im_name3]);
fname1=([keypoint_path,key_name1]) ;
save([fname1,'.key.mat']);
```

++

These scripts does the signature verification stage. When a signature claimed to be of a known signer is presented , its SIFT descriptors are extracted. A MATLAB template containing descriptors of the known signatures is loaded and the distance measures between its parameters and those of the claimed signatures are calculated . The intra-class and inter - class thresholds are compared to ascertain whether its genuine or not .

VERIFYsignatureUSINGMAX.m

This script uses the maximum intra - class distance as the threshold.

```
clear;
close all;
global EvansSigMsc;
Signa_path = [EvansSigMsc 'E:\SIGNATURE_EVANS'];
addpath( Signa_path );
im_path = [Signa_path,'/Signatures/'];
keypoint_path = [Signa_path,'/KEYPOINTS/'];
octaves = 4;
intervals = 2;
cache = 1;
claimedSignature= input('Enter the claimed signature \n','s');
```

```
im4=im2double(imreadbw([im_path,claimedSignature,'.png'])) ;
fprintf( 2, 'Computing keypoints for the claimed signatures.\n' );
[pos4,scale4,orient4,desc4]
=SIFT(im4,octaves,intervals,ones(size(im4)),0.02,10.0,2);
knowntemp= input('Enter the KNOWN WRITER TEMPLATE \n','s');
fname1=([keypoint_path,knowntemp]) ;
load([fname1,'.key.mat']);
fprintf( 2, 'RETRIEVE THE KNOWN WRITER TEMPLATE .\n' );
[ D41,D42,D43,AGD41,AGD42,AGD43,interMin,interMax,interAvg]
= interclassEuclidean( desc1,desc2,desc3,desc4);
if interMax < intraMax
  fprintf( 2, 'THE CLAIMED SIGNATURE IS GENUINE .\n' )
elseif interMax > intraMax
    fprintf( 2, 'THE CLAIMED SIGNATURE IS NOT GENUINE .\n' )
end
```

++

VERIFYsignatureUSINGAVG.m

This script uses average intra - class distance as the threshold.

```
clear;
close all;
global EvansSigMsc;
Signa_path = [EvansSigMsc 'E:\SIGNATURE_EVANS'];
addpath( Signa_path );
im_path = [Signa_path,'/Signatures/'];
keypoint_path = [Signa_path,'/KEYPOINTS/'];
octaves = 4;
intervals = 2;
cache = 1;
claimedSignature= input('Enter the claimed signature \n','s');
im4=im2double(imreadbw([im_path,claimedSignature,'.png'])) ;
fprintf( 2, 'Computing keypoints for the claimed signatures.\n' );
```

```
[pos4, scale4, orient4, desc4]
= SIFT( im4, octaves, intervals, ones(size(im4)), 0.02, 10.0, 2 );
knowntemp= input('Enter the KNOWN WRITER TEMPLATE \n','s');
fname1=([keypoint_path,knowntemp]) ;
load([fname1,'.key.mat']);
fprintf( 2, 'RETRIEVE THE KNOWN WRITER TEMPLATE .\n' );
[ D41,D42,D43,AGD41,AGD42,AGD43,interMin,interMax,interAvg]
 = interclassEuclidean( desc1,desc2,desc3,desc4);
if interAvg < intraAvg
  fprintf( 2, 'THE CLAIMED SIGNATURE IS GENUINE .\n' )
elseif interAvg > intraAvg
    fprintf( 2, 'THE CLAIMED SIGNATURE IS NOT GENUINE .\n' )
end
```

++

VERIFYsignatureUSINGMIN.m

This script uses minimum intra - class distance as the threshold.

```
clear;
close all;
global EvansSigMsc;
Signa_path = [EvansSigMsc 'E:\SIGNATURE_EVANS'];
addpath( Signa_path );
im_path = [Signa_path,'/Signatures/'];
keypoint_path = [Signa_path,'/KEYPOINTS/'];
octaves = 4;
intervals = 2;
cache = 1;
claimedSignature= input('Enter the claimed signature \n','s');
im4=im2double(imreadbw([im_path,claimedSignature,'.png'])) ;
fprintf( 2, 'Computing keypoints for the claimed signatures.\n' );
[pos4, scale4, orient4, desc4]
= SIFT( im4, octaves, intervals, ones(size(im4)), 0.02, 10.0, 2 );
knowntemp= input('Enter the KNOWN WRITER TEMPLATE \n','s');
```

```
fname1=([keypoint_path,knowntemp]) ;
load([fname1,'.key.mat']);
fprintf( 2, 'RETRIEVE THE KNOWN WRITER TEMPLATE .\n' );
[ D41,D42,D43,AGD41,AGD42,AGD43,interMin,interMax,interAvg]
= interclassEuclidean( desc1,desc2,desc3,desc4);
if interMax <=  maxRange
  fprintf( 2, 'THE CLAIMED SIGNATURE IS GENUINE .\n' )
elseif interMax => maxRange
    fprintf( 2, 'THE CLAIMED SIGNATURE IS NOT GENUINE .\n' )
end
```

++

VERIFYsignatureUSINGmaxRANGE.m

This script adds a distance of 0.05 above maximumm intra - class distance and use this as the threshold.

```
clear;
close all;
global EvansSigMsc;
Signa_path = [EvansSigMsc 'E:\SIGNATURE_EVANS'];
addpath( Signa_path );
im_path = [Signa_path,'/Signatures/'];
keypoint_path = [Signa_path,'/KEYPOINTS/'];
octaves = 4;
intervals = 2;
cache = 1;
claimedSignature= input('Enter the claimed signature \n','s');
im4=im2double(imreadbw([im_path,claimedSignature,'.png'])) ;
fprintf( 2, 'Computing keypoints for the claimed signatures.\n' );
[pos4, scale4, orient4, desc4]
 = SIFT( im4, octaves, intervals, ones(size(im4)), 0.02, 10.0, 2 );
knowntemp= input('Enter the KNOWN WRITER TEMPLATE \n','s');
fname1=([keypoint_path,knowntemp]) ;
load([fname1,'.key.mat']);
```

```
fprintf( 2, ’RETRIEVE THE KNOWN WRITER TEMPLATE .\n’ );
[ D41,D42,D43,AGD41,AGD42,AGD43,interMin,interMax,interAvg]
= interclassEuclidean( desc1,desc2,desc3,desc4);
if interMax <=  maxRange
  fprintf( 2, ’THE CLAIMED SIGNATURE IS GENUINE .\n’ )
elseif interMax => maxRange
    fprintf( 2, ’THE CLAIMED SIGNATURE IS NOT GENUINE .\n’ )
end
```

++

VERIFYsignatureUSINGminRANGE.m

This script substracts a distance of 0.05 from the minimum intra - class distance and use this as the threshold.

```
clear;
close all;
global EvansSigMsc;
Signa_path = [EvansSigMsc ’E:\SIGNATURE_EVANS’];
addpath( Signa_path );
im_path = [Signa_path,’/Signatures/’];
keypoint_path = [Signa_path,’/KEYPOINTS/’];
octaves = 4;
intervals = 2;
cache = 1;

claimedSignature= input(’Enter the claimed signature \n’,’s’);
im4=im2double(imreadbw([im_path,claimedSignature,’.png’])) ;
fprintf( 2, ’Computing keypoints for the claimed signatures.\n’ );
[pos4, scale4, orient4, desc4] =
 SIFT( im4, octaves, intervals, ones(size(im4)), 0.02, 10.0, 2 );
knowntemp= input(’Enter the KNOWN WRITER TEMPLATE \n’,’s’);
fname1=([keypoint_path,knowntemp]) ;
load([fname1,’.key.mat’]);
fprintf( 2, ’RETRIEVE THE KNOWN WRITER TEMPLATE .\n’ );
```

```
[ D41,D42,D43,AGD41,AGD42,AGD43,interMin,interMax,interAvg]
 = interclassEuclidean( desc1,desc2,desc3,desc4);
if interMin <=  minRange
  fprintf( 2, 'THE CLAIMED SIGNATURE IS GENUINE .\n' )
elseif interMin = > minRange
    fprintf( 2, 'THE CLAIMED SIGNATURE IS NOT GENUINE .\n' )
end
```

++

CREATETtesttemplate.m

This script loads all the images used for testing the accuracy of the verifier. It calls the functions to extract SIFT descriptors of the signature, calculates the distances , the class thresholds and the outlier detection.

All these parameters are stored as a matlab file that will be used to measure the accuracy of the verifier.

```
clear;
close all;
global EvansSigMsc;
Signa_path = [EvansSigMsc 'E:/SIGNATUREEVANS'];
addpath( Signa_path );
im_path = [Signa_path,'/Signatures/TESTSET/'];
keypoint_path = [Signa_path,'/KEYPOINTS/TESTKEYSET/'];
octaves = 4;
intervals = 2;
cache = 1;
% Load the test signatures.
% Extract their SIFT features
%calculate the Euclidean distances between the featuressig1=im2double(imrea
[pos1, scale1, orient1, desc1 ]
= SIFT( sig1, octaves, intervals, ones(size(sig1)), 0.02, 10.0, 1 );
sig2=im2double(imreadbw([im_path,'2.png'])) ;
[pos2, scale2, orient2, desc2]
```

```
= SIFT( sig2, octaves, intervals, ones(size(sig2)), 0.02, 10.0, 1 );

sig3=im2double(imreadbw([im_path,'3.png'])) ;
[pos3, scale3, orient3, desc3 ]
= SIFT( sig3, octaves, intervals, ones(size(sig3)), 0.02, 10.0, 1 );

sig4=im2double(imreadbw([im_path,'4.png'])) ;

[pos4, scale4, orient4, desc4 ]
= SIFT( sig4, octaves, intervals, ones(size(sig4)), 0.02, 10.0, 1 );

sig5=im2double(imreadbw([im_path,'5.png'])) ;
[pos5, scale5, orient5, desc5 ]
= SIFT( sig5, octaves, intervals, ones(size(sig5)), 0.02, 10.0, 1 );
sig6=im2double(imreadbw([im_path,'6.png'])) ;
[pos6, scale6, orient6, desc6 ] =
SIFT( sig6, octaves, intervals, ones(size(sig6)), 0.02, 10.0, 1 );
sig7=im2double(imreadbw([im_path,'7.png'])) ;
[pos7, scale7, orient7, desc7 ] =
SIFT( sig7, octaves, intervals, ones(size(sig7)), 0.02, 10.0, 1 );
sig8=im2double(imreadbw([im_path,'8.png'])) ;
[pos8, scale8, orient8, desc8 ] =
SIFT( sig8, octaves, intervals, ones(size(sig8)), 0.02, 10.0, 1 );
sig9=im2double(imreadbw([im_path,'9.png'])) ;
[pos9, scale9, orient9, desc9 ] =
SIFT( sig9, octaves, intervals, ones(size(sig9)), 0.02, 10.0, 1 );
sig10=im2double(imreadbw([im_path,'10.png'])) ;
[pos10, scale10, orient10, desc10 ] =
 SIFT( sig10, octaves, intervals, ones(size(sig10)), 0.02, 10.0, 1 );
sig11=im2double(imreadbw([im_path,'11.png'])) ;
[pos11, scale11, orient11, desc11 ] =
SIFT( sig11, octaves, intervals, ones(size(sig11)), 0.02, 10.0, 1 );
sig12=im2double(imreadbw([im_path,'12.png'])) ;
[pos12, scale12, orient12, desc12 ] =
```

```
SIFT( sig12, octaves, intervals, ones(size(sig12)), 0.02, 10.0, 1 );
sig13=im2double(imreadbw([im_path,'13.
png'])) ;
[pos13, scale13, orient13, desc13 ]
= SIFT( sig13, octaves, intervals, ones(size(sig13)), 0.02, 10.0, 1 );

sig14=im2double(imreadbw([im_path,'14.png'])) ;
[pos14, scale14, orient14, desc14 ]
= SIFT( sig14, octaves, intervals, ones(size(sig14)), 0.02, 10.0, 1 );
sig15=im2double(imreadbw([im_path,'15.png'])) ;
[pos15, scale15, orient15, desc15 ]
= SIFT( sig15, octaves, intervals, ones(size(sig15)), 0.02, 10.0, 1 );
sig16=im2double(imreadbw([im_path,'16.png'])) ;
[pos16, scale16, orient16, desc16 ]
= SIFT( sig16, octaves, intervals, ones(size(sig16)), 0.02, 10.0, 1 );

sig17=im2double(imreadbw([im_path,'17.png'])) ;
[pos17, scale17, orient17, desc17 ]
= SIFT( sig17, octaves, intervals, ones(size(sig17)), 0.02, 10.0, 1 );

sig18=im2double(imreadbw([im_path,'18.png'])) ;
[pos18, scale18, orient18, desc18 ]
= SIFT( sig18, octaves, intervals, ones(size(sig18)), 0.02, 10.0, 1 );

sig19=im2double(imreadbw([im_path,'19.png'])) ;
[pos19, scale19, orient19, desc19 ]
= SIFT( sig19, octaves, intervals, ones(size(sig19)), 0.02, 10.0, 1 );

sig20=im2double(imreadbw([im_path,'20.png'])) ;
[pos20, scale20, orient20, desc20]
= SIFT( sig20, octaves, intervals, ones(size(sig20)), 0.02, 10.0, 1 );

sig21=im2double(imreadbw([im_path,'21.png'])) ;
[pos21, scale21, orient21, desc21]
```

```
= SIFT( sig21, octaves, intervals, ones(size(sig21)), 0.02, 10.0, 1 );

sig22=im2double(imreadbw([im_path,'22.png'])) ;
[pos22, scale22, orient22, desc22]
= SIFT( sig22, octaves, intervals, ones(size(sig22)), 0.02, 10.0, 1 );

sig23=im2double(imreadbw([im_path,'23.png'])) ;
[pos23, scale23, orient23, desc23]
= SIFT( sig23, octaves, intervals, ones(size(sig23)), 0.02, 10.0, 1 );



sig24=im2double(imreadbw([im_path,'24.png'])) ;
[pos24, scale24, orient24, desc24]
= SIFT( sig24, octaves, intervals, ones(size(sig24)), 0.02, 10.0, 1 );

sig25=im2double(imreadbw([im_path,'25.png'])) ;
[pos25, scale25, orient25, desc25]
= SIFT( sig25, octaves, intervals, ones(size(sig25)), 0.02, 10.0, 1 );

sig26=im2double(imreadbw([im_path,'26.png'])) ;
[pos26, scale26, orient26, desc26 ]
= SIFT( sig26, octaves, intervals, ones(size(sig26)), 0.02, 10.0, 1 );

sig27=im2double(imreadbw([im_path,'27.png'])) ;
[pos27, scale27, orient27, desc27 ]
= SIFT( sig27, octaves, intervals, ones(size(sig27)), 0.02, 10.0, 1 );

sig28=im2double(imreadbw([im_path,'28.png'])) ;
[pos28, scale28, orient28, desc28 ]
= SIFT( sig28, octaves, intervals, ones(size(sig28)), 0.02, 10.0, 1 );

sig29=im2double(imreadbw([im_path,'29.png'])) ;
[pos29, scale29, orient29, desc29 ]
```

```
= SIFT( sig29, octaves, intervals, ones(size(sig29)), 0.02, 10.0, 1 );

sig30=im2double(imreadbw([im_path,'30.png'])) ;
[pos30, scale30, orient30, desc30]
= SIFT( sig30, octaves, intervals, ones(size(sig30)), 0.02, 10.0, 1 );


sig31=im2double(imreadbw([im_path,'31.png'])) ;
[pos31, scale31, orient31, desc31]
= SIFT( sig31, octaves, intervals, ones(size(sig31)), 0.02, 10.0, 1 );

sig32=im2double(imreadbw([im_path,'32.png'])) ;
[pos32, scale32, orient32, desc32]
= SIFT( sig32, octaves, intervals, ones(size(sig32)), 0.02, 10.0, 1 );

sig33=im2double(imreadbw([im_path,'33.png'])) ;
[pos33, scale33, orient33, desc33]
= SIFT( sig33, octaves, intervals, ones(size(sig33)), 0.02, 10.0, 1 );

sig34=im2double(imreadbw([im_path,'34.png'])) ;
[pos34, scale34, orient34, desc34]
 = SIFT( sig34, octaves, intervals, ones(size(sig34)), 0.02, 10.0, 1 );

sig35=im2double(imreadbw([im_path,'35.png'])) ;
[pos35, scale35, orient35, desc35]
= SIFT( sig35, octaves, intervals, ones(size(sig35)), 0.02, 10.0, 1 );

sig36=im2double(imreadbw([im_path,'36.png'])) ;
[pos36, scale36, orient36, desc36 ]
= SIFT( sig36, octaves, intervals, ones(size(sig36)), 0.02, 10.0, 1 );

sig37=im2double(imreadbw([im_path,'37.png'])) ;
[pos37, scale37, orient37, desc37 ]
= SIFT( sig37, octaves, intervals, ones(size(sig37)), 0.02, 10.0, 1 );
```

```
sig38=im2double(imreadbw([im_path,'38.png'])) ;
[pos38, scale38, orient38, desc38 ]
= SIFT( sig38, octaves, intervals, ones(size(sig38)), 0.02, 10.0, 1 );

sig39=im2double(imreadbw([im_path,'39.png'])) ;
[pos39, scale39, orient39, desc39 ]
= SIFT( sig39, octaves, intervals, ones(size(sig39)), 0.02, 10.0, 1 );

sig40=im2double(imreadbw([im_path,'40.png'])) ;
[pos40, scale40, orient40, desc40]
= SIFT( sig40, octaves, intervals, ones(size(sig40)), 0.02, 10.0, 1 );

sig41=im2double(imreadbw([im_path,'41.png'])) ;
[pos41, scale41, orient41, desc41]
= SIFT( sig41, octaves, intervals, ones(size(sig41)), 0.02, 10.0, 1 );

sig42=im2double(imreadbw([im_path,'42.png'])) ;
[pos42, scale42, orient42, desc42]
= SIFT( sig42, octaves, intervals, ones(size(sig42)), 0.02, 10.0, 1 );

sig43=im2double(imreadbw([im_path,'43.png'])) ;
[pos43, scale43, orient43, desc43]
= SIFT( sig43, octaves, intervals, ones(size(sig43)), 0.02, 10.0, 1 );

sig44=im2double(imreadbw([im_path,'44.png'])) ;
[pos44, scale44, orient44, desc44]
= SIFT( sig44, octaves, intervals, ones(size(sig44)), 0.02, 10.0, 1 );

sig45=im2double(imreadbw([im_path,'45.png'])) ;
[pos45, scale45, orient45, desc45]
= SIFT( sig45, octaves, intervals, ones(size(sig45)), 0.02, 10.0, 1 );

sig46=im2double(imreadbw([im_path,'46.png'])) ;
```

```
[pos46, scale46, orient46, desc46 ] =
SIFT( sig46, octaves, intervals, ones(size(sig46)), 0.02, 10.0, 1 );

sig47=im2double(imreadbw([im_path,'47.png'])) ;
[pos47, scale47, orient47, desc47 ]
= SIFT( sig47, octaves, intervals, ones(size(sig47)), 0.02, 10.0, 1 );

sig48=im2double(imreadbw([im_path,'48.png'])) ;
[pos48, scale48, orient48, desc48 ]
= SIFT( sig48, octaves, intervals, ones(size(sig48)), 0.02, 10.0, 1 );

sig49=im2double(imreadbw([im_path,'49.png'])) ;
[pos49, scale49, orient49, desc49 ]
= SIFT( sig49, octaves, intervals, ones(size(sig49)), 0.02, 10.0, 1 );

sig50=im2double(imreadbw([im_path,'50.png'])) ;
[pos50, scale50, orient50, desc50]
 = SIFT( sig50, octaves, intervals, ones(size(sig50)), 0.02, 10.0, 1 );

sig51=im2double(imreadbw([im_path,'51.png'])) ;
[pos51, scale51, orient51, desc51]
= SIFT( sig51, octaves, intervals, ones(size(sig51)), 0.02, 10.0, 1 );

sig52=im2double(imreadbw([im_path,'
52.png'])) ;
[pos52, scale52, orient52, desc52]
= SIFT( sig52, octaves, intervals, ones(size(sig52)), 0.02, 10.0, 1 );

sig53=im2double(imreadbw([im_path,'53.png'])) ;
[pos53, scale53, orient53, desc53]
= SIFT( sig53, octaves, intervals, ones(size(sig53)), 0.02, 10.0, 1 );

sig54=im2double(imreadbw([im_path,'54.png'])) ;
[pos54, scale54, orient54, desc54]
```

```
= SIFT( sig54, octaves, intervals, ones(size(sig54)), 0.02, 10.0, 1 );

sig55=im2double(imreadbw([im_path,'55.png'])) ;
[pos55, scale55, orient55, desc55]
= SIFT( sig55, octaves, intervals, ones(size(sig55)), 0.02, 10.0, 1 );

sig56=im2double(imreadbw([im_path,'56.png'])) ;
[pos56, scale56, orient56, desc56 ]
= SIFT( sig56, octaves, intervals, ones(size(sig56)), 0.02, 10.0, 1 );

sig57=im2double(imreadbw([im_path,'57.png'])) ;
[pos57, scale57, orient57, desc57 ]
= SIFT( sig57, octaves, intervals, ones(size(sig57)), 0.02, 10.0, 1 );

sig58=im2double(imreadbw([im_path,'58.png'])) ;
[pos58, scale58, orient58, desc58 ]
= SIFT( sig58, octaves, intervals, ones(size(sig58)), 0.02, 10.0, 1 );

sig59=im2double(imreadbw([im_path,'59.png'])) ;
[pos59, scale59, orient59, desc59 ]
= SIFT( sig59, octaves, intervals, ones(size(sig59)), 0.02, 10.0, 1 );

sig60=im2double(imreadbw([im_path,'60.png'])) ;
[pos60, scale60, orient60, desc60]
= SIFT( sig60, octaves, intervals, ones(size(sig60)), 0.02, 10.0, 1 );

sig61=im2double(imreadbw([im_path,'61.png'])) ;
[pos61, scale61, orient61, desc61]
= SIFT( sig61, octaves, intervals, ones(size(sig61)), 0.02, 10.0, 1 );


sig62=im2double(imreadbw([im_path,'62.png'])) ;
[pos62, scale62, orient62, desc62]
= SIFT( sig62, octaves, intervals, ones(size(sig62)), 0.02, 10.0, 1 );
```

```
sig63=im2double(imreadbw([im_path,'63.png'])) ;
[pos63, scale63, orient63, desc63]
= SIFT( sig63, octaves, intervals, ones(size(sig63)), 0.02, 10.0, 1 );

sig64=im2double(imreadbw([im_path,'64.png'])) ;
[pos64, scale64, orient64, desc64]
= SIFT( sig64, octaves, intervals, ones(size(sig64)), 0.02, 10.0, 1 );

sig65=im2double(imreadbw([im_path,'65.png'])) ;
[pos65, scale65, orient65, desc65] =
 SIFT( sig65, octaves, intervals, ones(size(sig65)), 0.02, 10.0, 1 );

sig66=im2double(imreadbw([im_path,'66.png'])) ;
[pos66, scale66, orient66, desc66 ]
= SIFT( sig66, octaves, intervals, ones(size(sig66)), 0.02, 10.0, 1 );

sig67=im2double(imreadbw([im_path,'67.png'])) ;
[pos67, scale67, orient67, desc67 ]
= SIFT( sig67, octaves, intervals, ones(size(sig67)), 0.02, 10.0, 1 );

sig68=im2double(imreadbw([im_path,'68.png'])) ;
[pos68, scale68, orient68, desc68 ]
= SIFT( sig68, octaves, intervals, ones(size(sig68)), 0.02, 10.0, 1 );

sig69=im2double(imreadbw([im_path,'69.png'])) ;
[pos69, scale69, orient69, desc69 ]
= SIFT( sig69, octaves, intervals, ones(size(sig69)), 0.02, 10.0, 1 );

sig70=im2double(imreadbw([im_path,'70.png'])) ;
[pos70, scale70, orient70, desc70]
= SIFT( sig70, octaves, intervals, ones(size(sig70)), 0.02, 10.0, 1 );

sig71=im2double(imreadbw([im_path,'71.png'])) ;
```

```
[pos71, scale71, orient71, desc71]
= SIFT( sig71, octaves, intervals, ones(size(sig71)), 0.02, 10.0, 1 );
sig72=im2double(imreadbw([im_path,'72.png'])) ;
[pos72, scale72, orient72, desc72]
= SIFT( sig72, octaves, intervals, ones(size(sig72)), 0.02, 10.0, 1 );

sig73=im2double(imreadbw([im_path,'73.png'])) ;
[pos73, scale73, orient73, desc73] =
 SIFT( sig73, octaves, intervals, ones(size(sig73)), 0.02, 10.0, 1 );

sig74=im2double(imreadbw([im_path,'74.png'])) ;
[pos74, scale74, orient74, desc74] =
 SIFT( sig74, octaves, intervals, ones(size(sig74)), 0.02, 10.0, 1 );

sig75=im2double(imreadbw([im_path,'75.png'])) ;
[pos75, scale75, orient75, desc75]
= SIFT( sig75, octaves, intervals, ones(size(sig75)), 0.02, 10.0, 1 );

sig76=im2double(imreadbw([im_path,'76.png'])) ;
[pos76, scale76, orient76, desc76 ]
= SIFT( sig76, octaves, intervals, ones(size(sig76)), 0.02, 10.0, 1 );

sig77=im2double(imreadbw([im_path,'77.png'])) ;
[pos77, scale77, orient77, desc77 ]
 = SIFT( sig77, octaves, intervals, ones(size(sig77)), 0.02, 10.0, 1 );

sig78=im2double(imreadbw([im_path,'78.png'])) ;
[pos78, scale78, orient78, desc78 ]
= SIFT( sig78, octaves, intervals, ones(size(sig78)), 0.02, 10.0, 1 );

sig79=im2double(imreadbw([im_path,'79.png'])) ;
[pos79, scale79, orient79, desc79 ]
= SIFT( sig79, octaves, intervals, ones(size(sig79)), 0.02, 10.0, 1 );
```

```
sig80=im2double(imreadbw([im_path,'80.png'])) ;
[pos80, scale80, orient80, desc80]
= SIFT( sig80, octaves, intervals, ones(size(sig80)), 0.02, 10.0, 1 );

sig81=im2double(imreadbw([im_path,'81.png'])) ;
[pos81, scale81, orient81, desc81]
= SIFT( sig81, octaves, intervals, ones(size(sig81)), 0.02, 10.0, 1 );


sig82=im2double(imreadbw([im_path,'82.png'])) ;
[pos82, scale82, orient82, desc82]
= SIFT( sig82, octaves, intervals, ones(size(sig82)), 0.02, 10.0, 1 );

sig83=im2double(imreadbw([im_path,'83.png'])) ;
[pos83, scale83, orient83, desc83]
= SIFT( sig83, octaves, intervals, ones(size(sig83)), 0.02, 10.0, 1 );

sig84=im2double(imreadbw([im_path,'84.png'])) ;
[pos84, scale84, orient84, desc84]
= SIFT( sig84, octaves, intervals, ones(size(sig84)), 0.02, 10.0, 1 );

sig85=im2double(imreadbw([im_path,'85.png'])) ;
[pos85, scale85, orient85, desc85]
= SIFT( sig85, octaves, intervals, ones(size(sig85)), 0.02, 10.0, 1 );

sig86=im2double(imreadbw([im_path,'86.png'])) ;
[pos86, scale86, orient86, desc86 ]
= SIFT( sig86, octaves, intervals, ones(size(sig86)), 0.02, 10.0, 1 );

sig87=im2double(imreadbw([im_path,'87.png'])) ;
[pos87, scale87, orient87, desc87 ]
= SIFT( sig87, octaves, intervals, ones(size(sig87)), 0.02, 10.0, 1 );

sig88=im2double(imreadbw([im_path,'88.png'])) ;
```

```
[pos88, scale88, orient88, desc88 ]
= SIFT( sig88, octaves, intervals, ones(size(sig88)), 0.02, 10.0, 1 );

sig89=im2double(imreadbw([im_path,'89.png'])) ;
[pos89, scale89, orient89, desc89 ]
= SIFT( sig89, octaves, intervals, ones(size(sig89)), 0.02, 10.0, 1 );

sig90=im2double(imreadbw([im_path,'90.png'])) ;
[pos90, scale90, orient90, desc90]
= SIFT( sig90, octaves, intervals, ones(size(sig90)), 0.02, 10.0, 1 );


[ D13,D23,D12,AGD12,AGD13,AGD23,intraMin1,
intraMax1,intraAvg1,
maxRange1,minRange1] = intra1( desc1,desc2,desc3);
[ D43,D42,D41,AGD41,AGD43,AGD42,interMin1a
,interMax1a,interAvg1a]
= inter1a(desc4, desc1,desc2,desc3);
[ D53,D52,D51,AGD51,AGD53,AGD52,interMin1b
interMax1b,interAvg1b] = inter1b(desc5, desc1,desc2,desc3);
[ D68,D78,D67,AGD68,AGD78,AGD67,intraMin2,
intraMax2,intraAvg2,maxRange2,minRange2]
= intra2( desc6,desc7,desc8);
[ D69,D79,D89,AGD69,AGD79,AGD89,interMax2a,
interMin2a,interAvg2a]
= inter2a(desc9, desc6,desc7,desc8);
[ D610,D710,D810,AGD610,AGD710,
AGD810,interMax2b,interMin2b,interAvg2b]
= inter2b(desc10, desc6,desc7,desc8);
[ D1113,D1213,D1112,AGD1112,AGD1113,
AGD1213,intraMin3,intraMax3,intraAvg3,
maxRange3,minRange3] = intra3( desc11,desc12,desc13);
[ D1114,D1214,D1314,AGD1114,AGD1214,AGD1314,
interMin3a,interMax3a,interAvg3a]
```

```
= inter3a( desc14,desc11,desc12,desc13);
[ D1115,D1215,D1315,AGD1115,AGD1215,
AGD1315,interMin3b,interMax3b,interAvg3b]
= inter3b( desc15,desc11,desc12,desc13);
[ D1618,D1718,D1617,AGD1618,AGD1718,AGD1617,intraMin4,
intraMax4,intraAvg4,maxRange4,minRange4]
= intra4( desc16,desc17,desc18);
[ D1619,D1719,D1819,AGD1619,AGD1719
,AGD1819,interMin4a,interMax4a,interAvg4a]
= inter4a( desc19,desc16,desc17,desc18);
[ D1620,D1720,D1820,AGD1620,
AGD1720,AGD1820,interMin4b,interMax4b,interAvg4b]
 =inter4b( desc20,desc16,desc17,desc18);
[ D2123,D2223,D2122,AGD2122,AGD2123,AGD2223,
intraMin5,intraMax5,intraAvg5,maxRange5,minRange5]
 = intra5( desc21,desc22,desc23);
[ D2124,D2224,D2324,AGD2124,AGD2224,AGD2324,
interMin5a,interMax5a,interAvg5a]
= inter5a( desc24,desc21,desc22,desc23);
[ D2125,D2225,D2325,AGD2125,AGD2225,AGD2325,
interMin5b,interMax5b,interAvg5b]
= inter5b( desc25,desc21,desc22,desc23);
[ D2628,D2728,D2627,AGD2628,AGD2728,AGD2627,
intraMin6,intraMax6,intraAvg6,
maxRange6,minRange6] = intra6( desc26,desc27,desc28);
[ D2629,D2729,D2829,AGD2629,AGD2729,AGD2829,
interMin6a,interMax6a,interAvg6a]
= inter6a( desc29,desc26,desc27,desc28);
[ D2630,D2730,D2830,AGD2630,AGD2730,
AGD2830,interMin6b,interMax6b,interAvg6b]
= inter6b( desc30,desc26,desc27,desc28);
[ D3133,D3233,D3132,AGD3132,AGD3133,AGD3233
,intraMin7,intraMax7,intraAvg7,maxRange7,minRange7]
 = intra7( desc31,desc32,desc33);
```

```
[ D3134,D3234,D3334,AGD3134,AGD3234,AGD3334,
interMin7a,interMax7a,interAvg7a]
= inter7a( desc34,desc31,desc32,desc33);
[ D3135,D3235,D3335,AGD3135,AGD3235,AGD3335,
interMin7b,interMax7b,interAvg7b]
= inter7b( desc35,desc31,desc32,desc33);
[ D3638,D3738,D3637,AGD3638,AGD3738,AGD3637
,intraMin8,intraMax8,intraAvg8,maxRange8,minRange8
 = intra8( desc36,desc37,desc38);
[ D3639,D3739,D3839,AGD3639,AGD3739,AGD3839,
interMin8a,interMax8a,interAvg8a]
= inter8a( desc39,desc36,desc37,desc38);
[ D3640,D3740,D3840,AGD3640,AGD3740,AGD3840,
interMin8b,interMax8b,interAvg8b]
= inter8b( desc40,desc36,desc37,desc38);
[ D4143,D4243,D4142,AGD4142,AGD4143,AGD4243
,intraMin9,intraMax9,intraAvg9,maxRange9,minRange9]
 = intra9( desc41,desc42,desc43);
[ D4144,D4244,D4344,AGD4144,AGD4244,AGD4344,
interMin9a,interMax9a,interAvg9a]
= inter9a( desc44,desc41,desc42,desc43);
[ D4145,D4245,D4345,AGD4145,AGD4245,AGD4345,
interMin9b,interMax9b,interAvg9b]
= inter9b( desc45,desc41,desc42,desc43);
[ D4648,D4748,D4647,AGD4648,AGD4748,AGD4647,
intraMin10,intraMax10,intraAvg10,maxRange10,minRange10]
 = intra10( desc46,desc47,desc48);
[ D4649,D4749,D4849,AGD4649,AGD4749,AGD4849,
interMin10a,interMax10a,interAvg10a]
= inter10a( desc49,desc46,desc47,desc48);
[ D4650,D4750,D4850,AGD4650,AGD4750,AGD4850,i
nterMin10b,interMax10b,interAvg10b]
= inter10b( desc50,desc46,desc47,desc48);
[ D5153,D5253,D5152,AGD5152,AGD5153,AGD5253,
```

```
intraMin11,intraMax11,intraAvg11,maxRange11,minRange11]
 = intra11( desc51,desc52,desc53);
[ D5154,D5254,D5354,AGD5154,AGD5254,AGD5354,
interMin11a,interMax11a,interAvg11a]
 = inter11a( desc54,desc51,desc52,desc53);
[ D5155,D5255,D5355,AGD5155,AGD5255,AGD5355,
interMin11b,interMax11b,interAvg11b]
= inter11b( desc55,desc51,desc52,desc53);
[ D5658,D5758,D5657,AGD5658,
AGD5758,AGD5657,intraMin12,intraMax12,intraAvg12,maxRange12,minRange12]
 = intra12( desc56,desc57,desc58);
[ D5659,D5759,D5859,AGD5659,AGD5759,AGD5859,i
nterMin12a,interMax12a,interAvg12a]
= inter12a( desc59,desc56,desc57,desc58);
[ D5660,D5760,D5860,AGD5660,AGD5760,AGD5860,
interMin12b,interMax12b,interAvg12b]
= inter12b( desc60,desc56,desc57,desc58);
[ D6163,D6263,D6162,AGD6162,AGD6163,AGD6263,
intraMin13,intraMax13,
intraAvg13,maxRange13,minRange13]
= intra13( desc61,desc62,desc63);
[ D6164,D6264,D6364,AGD6164,AGD6264,AGD6364,
interMin13a,interMax13a,interAvg13a]
= inter13a(desc64, desc61,desc62,desc63);
[ D6165,D6265,D6365,AGD6165,AGD6265,AGD6365,
interMin13b,interMax13b,interAvg13b]
= inter13b(desc65, desc61,desc62,desc63);
[ D6668,D6768,D6667,AGD6768,AGD6668,AGD6667,
intraMin14,intraMax14,intraAvg14,
maxRange14,minRange14] = intra14( desc66,desc67,desc68);
[ D6669,D6769,D6869,AGD6669,AGD6769,AGD6869,
interMin14a,interMax14a,
interAvg14a] = inter14a( desc69,desc66,desc67,desc68);
[ D6670,D6770,D6870,AGD6670,AGD6770,AGD6870,interMin14b,
```

```
interMax14b,interAvg14b] = inter14b( desc70,desc66,desc67,desc68);
[ D7173,D7273,D7172,AGD7172,AGD7173,AGD7273,intraMin15
,intraMax15,intraAvg15,maxRange15,
minRange15] = intra15( desc71,desc72,desc73);
[ D7174,D7274,D7374,AGD7174,AGD7274,AGD7374,interMin15a
,interMax15a,interAvg15a] = inter15a( desc74,desc71,desc72,desc73);
[ D7175,D7275,D7375,AGD7175,AGD7275,AGD7375,
interMin15b,interMax15b,interAvg15b] = inter15b( desc75,desc71,desc72,desc7
[ D7678,D7778,D7677,AGD7678,AGD7778,AGD7677,
intraMin16,intraMax16,intraAvg16,
maxRange16,minRange16] = intra16( desc76,desc77,desc78);
[ D7679,D7779,D7879,AGD7679,AGD7779,
AGD7879,interMin16a,interMax16a,interAvg16a]
= inter16a( desc79,desc76,desc77,desc78);
[ D7680,D7780,D7880,AGD7680,AGD7780,AGD7880
,interMin16b,interMax16b,interAvg16b]
= inter16b( desc80,desc76,desc77,desc78);

[ D8183,D8283,D8182,AGD8182,AGD8183,
AGD8283,intraMin17,intraMax17,intraAvg17,maxRange17,minRange17] =
intra17( desc81,desc82,desc83);
[ D8184,D8284,D8384,AGD8184,AGD8284,AGD8384,
interMin17a,interMax17a,interAvg17a]
= inter17a(desc84, desc81,desc82,desc83);
[ D8185,D8285,D8385,AGD8185,AGD8285,AGD8385,
interMin17b,interMax17b,interAvg17b]
= inter17b(desc85,desc81,desc82,desc83);
[ D8688,D8788,D8687,AGD8688,AGD8788,AGD8687,
intraMin18,intraMax18,intraAvg18,
maxRange18,minRange18] = intra18( desc86,desc87,desc88);
[ D8689,D8789,D8889,AGD8689,AGD8789,AGD8889,
interMin18a,interMax18a,interAvg18a]
= inter18a( desc89,desc86,desc87,desc88);
[ D8690,D8790,D8890,AGD8690,AGD8790,AGD8890,
```

```
interMin18b,interMax18b,interAvg18b]
= inter18b( desc90,desc86,desc87,desc88);
fprintf( 2, 'creating the test sample keypoints MATLAB template .\n' );
save([keypoint_path,'TESTSIGNATURES.mat']);
```

++

VERIFIERaccuracy.m

This script calculates the sensitivity and the specificity of the signature verfifier based on the test sample created by *CREATETtesttemplate.m* script.

```
load(['E:\SIGNATURE_EVANS\KEYPOINTS\TESTKEYSET\TESTSIGNATURES.mat']);

load(['C:\Documents and Settings\!-UseR\Desktop\SIGNATURE_EVANS\KEYPOINTS\T
if interMax1a < intraMax1
    maxtP1=1 ;
else maxtP1=0;
end
if interMax1a > intraMax1
    maxfN1=1;
else maxfN1=0;
end
if interAvg1a < intraAvg1
    avgtP1=1;
else  avgtP1=0;
end
if  interAvg1a > intraAvg1
    avgfN1=1;
else avgfN1=0;
end
if interMin1a < intraMin1
    mintP1=1;
else mintP1=0;
end
if interMin1a > intraMin1
    minfN1=1;
else minfN1=0;
end
if interMax1a < maxRange1
    maxRtP1=1;
```

```
else maxRtP1=0;
 end
if interMax1a > maxRange1
    maxRfN1=1;
else maxRfN1=0;
  end
if interMin1a < minRange1
    minRtP1=1;
else minRtP1=0;

end

if interMin1a > minRange1
    minRfN1=1;
else minRfN1=0;

end

if interMax1b < intraMax1
    maxfP1=1 ;
else maxfP1=0;
end
if interMax1b > intraMax1
    maxtN1=1;
else maxtN1=0;
end
if interAvg1b < intraAvg1
    avgfP1=1;
else  avgfP1=0;
end
if  interAvg1b > intraAvg1
    avgtN1=1;
else avgtN1=0;
end
```

```
if interMin1b < intraMin1
    minfP1=1;
else minfP1=0;
end

if interMin1b > intraMin1
    mintN1=1;
else mintN1=0;
end

if interMax1b < maxRange1
    maxRfP1=1;
else maxRfP1=0;
 end
if interMax1b > maxRange1
    maxRtN1=1;
else maxRtN1=0;
 end


if interMin1b < minRange1
    minRfP1=1;
else minRfP1=0;

end

if interMin1b > minRange1
    minRtN1=1;
else minRtN1=0;

end
if interMax2a < intraMax2
    maxtP2=1 ;
else maxtP2=0;
```

```
end
if interMax2a > intraMax2
    maxfN2=1;
else maxfN2=0;
end
if interAvg2a < intraAvg2
    avgtP2=1;
else  avgtP2=0;
end
if  interAvg2a > intraAvg2
    avgfN2=1;
else avgfN2=0;
end
if interMin2a < intraMin2
    mintP2=1;
else mintP2=0;
end

if interMin2a > intraMin2
    minfN2=1;
else minfN2=0;
end

if interMax2a < maxRange2
    maxRtP2=1;
else maxRtP2=0;

end
if interMax2a > maxRange2
    maxRfN2=1;
else maxRfN2=0;
 end
if interMin2a < minRange2
    minRtP2=1;
```

```
else minRtP2=0;
  end
if interMin2a > minRange2
    minRfN2=1;
else minRfN2=0;

end

if interMax2b < intraMax2
    maxfP2=1 ;
else maxfP2=0;
end

if interMax2b > intraMax2
    maxtN2=1;
else maxtN2=0;
end

if interAvg2b < intraAvg2
    avgfP2=1;
else  avgfP2=0;
end
if  interAvg2b > intraAvg2
    avgtN2=1;
else avgtN2=0;
end
if interMin2b < intraMin2
    minfP2=1;
else minfP2=0;
end

if interMin2b > intraMin2
    mintN2=1;
else mintN2=0;
```

```
end

if interMax2b < maxRange2
    maxRfP2=1;
else maxRfP2=0;

end
if interMax2b > maxRange2
    maxRtN2=1;
else maxRtN2=0;

end
if interMin2b < minRange2
    minRfP2=1;
else minRfP2=0;

end

if interMin2b > minRange2
    minRtN2=1;
else minRtN2=0;

end
if interMax3a < intraMax3
    maxtP3=1 ;
else maxtP3=0;
end
if interMax3a > intraMax3
    maxfN3=1;
else maxfN3=0;
end
if interAvg3a < intraAvg3
    avgtP3=1;
else  avgtP3=0;
```

```
end
if  interAvg3a > intraAvg3
    avgfN3=1;
else avgfN3=0;
end
if interMin3a < intraMin3
    mintP3=1;
else mintP3=0;
end
if interMin3a > intraMin3
    minfN3=1;
else minfN3=0;
end
if interMax3a < maxRange3
    maxRtP3=1;
else maxRtP3=0;

end
if interMax3a > maxRange3
    maxRfN3=1;
else maxRfN3=0;

end

if interMin3a < minRange3
    minRtP3=1;
else minRtP3=0;
 end
if interMin3a > minRange3
    minRfN3=1;
else minRfN3=0;
 end

if interMax3b < intraMax3
```

```
    maxfP3=1 ;
else maxfP3=0;
end

if interMax3b > intraMax3
    maxtN1=1;
else maxtN3=0;
end
if interAvg3b < intraAvg3
    avgfP3=1;
else  avgfP3=0;
end
if  interAvg3b > intraAvg3
    avgtN3=1;
else avgtN3=0;
end
if interMin3b < intraMin3
    minfP3=1;
else minfP3=0;
end

if interMin3b > intraMin3
    mintN3=1;
else mintN3=0;
end

if interMax3b < maxRange3
    maxRfP3=1;
else maxRfP3=0;

end

if interMax3b > maxRange3
    maxRtN3=1;
```

```
else maxRtN3=0;

end
if interMin3b < minRange3
    minRfP3=1;
else minRfP3=0;

end
if interMin3b > minRange3
    minRtN3=1;
else minRtN3=0;

end
if interMax4a < intraMax4
    maxtP4=1 ;
else maxtP4=0;
end
if interMax4a > intraMax4
    maxfN4=1;
else maxfN4=0;
end
if interAvg4a < intraAvg4
    avgtP4=1;
else  avgtP4=0;
end
if  interAvg4a > intraAvg4
    avgfN4=1;
else avgfN4=0;
end
if interMin4a < intraMin4
    mintP4=1;
else mintP4=0;
end
```

```
if interMin4a > intraMin4
    minfN4=1;
else minfN4=0;
end
if interMax4a < maxRange4
    maxRtP4=1;
else maxRtP4=0;
 end
if interMax4a > maxRange4
    maxRfN4=1;
else maxRfN4=0;

end

if interMin4a < minRange4
    minRtP4=1;
else minRtP4=0;
 end
if interMin4a > minRange4
    minRfN4=1;
else minRfN4=0;

end

if interMax4b < intraMax4
    maxfP4=1 ;
else maxfP4=0;
end

if interMax4b > intraMax4
    maxtN4=1;
else maxtN4=0;
end
if interAvg4b < intraAvg4
```

```
    avgfP4=1;
else  avgfP4=0;
end
if  interAvg4b > intraAvg4
    avgtN4=1;
else avgtN4=0;
end
if interMin4b < intraMin4
    minfP4=1;
else minfP4=0;
end

if interMin4b > intraMin4
    mintN4=1;
else mintN4=0;
end
if interMax4b < maxRange4
    maxRfP4=1;
else maxRfP4=0;
 end

if interMax4b > maxRange4
    maxRtN4=1;
else maxRtN4=0;

end

if interMin4b < minRange4
    minRfP4=1;
else minRfP4=0;
 end
if interMin4b > minRange4
    minRtN4=1;
else minRtN4=0;
```

```
end

if interMax5a < intraMax5
    maxtP5=1 ;
else maxtP5=0;
end
if interMax5a > intraMax5
    maxfN5=1;
else maxfN5=0;
end
if interAvg5a < intraAvg5
    avgtP5=1;
else  avgtP5=0;
end
if  interAvg5a > intraAvg5
    avgfN5=1;
else avgfN5=0;
end
if interMin5a < intraMin5
    mintP5=1;
else mintP5=0;
end
if interMin5a > intraMin5
    minfN5=1;
else minfN5=0;
end

if interMax5a < maxRange5
    maxRtP5=1;
else maxRtP5=0;
 end
if interMax5a > maxRange5
    maxRfN5=1;
```

```
else maxRfN5=0;
 end

if interMin5a < minRange5
    minRtP5=1;
else minRtP5=0;

end
if interMin5a > minRange5
    minRfN5=1;
else minRfN5=0;

end

if interMax5b < intraMax5
    maxfP5=1 ;
else maxfP5=0;
end
if interMax5b > intraMax5
    maxtN5=1;
else maxtN5=0;
end
if interAvg5b < intraAvg5
    avgfP5=1;
else  avgfP5=0;
end
if  interAvg5b > intraAvg5
    avgtN5=1;
else avgtN5=0;
end
if interMin5b < intraMin5
    minfP5=1;
else minfP5=0;
end
```

```
if interMin5b > intraMin5
    mintN5=1;
else mintN5=0;
end

if interMax5b < maxRange5
    maxRfP5=1;
else maxRfP5=0;
 end
if interMax5b > maxRange5
    maxRtN5=1;
else maxRtN5=0;

end
if interMin5b < minRange5
    minRfP5=1;
else minRfP5=0;
end
if interMin5b > minRange5
    minRtN5=1;
else minRtN5=0;

end

if interMax6a < intraMax6
    maxtP6=1 ;
else maxtP6=0;
end
if interMax6a > intraMax6
    maxfN6=1;
else maxfN6=0;
end

if interAvg6a < intraAvg6
```

```
    avgtP6=1;
else  avgtP6=0;
end
if  interAvg6a > intraAvg6
    avgfN6=1;
else avgfN6=0;
end
if interMin6a < intraMin6
    mintP6=1;
else mintP6=0;
end

if interMin6a > intraMin6
    minfN6=1;
else minfN6=0;
end

if interMax6a < maxRange6
    maxRtP6=1;
else maxRtP6=0;

end

if interMin6a > maxRange6
    maxRfN6=1;
else maxRfN6=0;

end

if interMin6a < minRange6
    minRtP6=1;
else minRtP6=0;

end
```

```
if interMax6a > minRange6
    minRfN6=1;
else minRfN6=0;

end

if interMax6b < intraMax6
    maxfP6=1 ;
else maxfP6=0;
end

if interMax6b > intraMax6
    maxtN6=1;
else maxtN6=0;
end

if interAvg6b < intraAvg6
    avgfP6=1;
else  avgfP6=0;
end
if  interAvg6b > intraAvg6
    avgtN6=1;
else avgtN6=0;
end
if interMin6b < intraMin6
    minfP6=1;
else minfP6=0;
end

if interMin6b > intraMin6
    mintN6=1;
else mintN6=0;
end
```

```
if interMax6b < maxRange6
    maxRfP6=1;
else maxRfP6=0;

end

if interMax6b > maxRange6
    maxRtN6=1;
else maxRtN6=0;

end


if interMin6b < minRange6
    minRfP6=1;
else minRfP6=0;

end

if interMin6b > minRange6
    minRtN6=1;
else minRtN6=0;

end
if interMax7a < intraMax7
    maxtP7=1 ;
else maxtP7=0;
end


if interMax7a > intraMax7
    maxfN7=1;
else maxfN7=0;
```

```
end
if interAvg7a < intraAvg7
    avgtP7=1;
else  avgtP7=0;
end
if  interAvg7a > intraAvg7
    avgfN7=1;
else avgfN7=0;
end
if interMin7a < intraMin7
    mintP7=1;
else mintP7=0;
end

if interMin7a > intraMin7
    minfN7=1;
else minfN7=0;
end
if interMax7a < maxRange7
    maxRtP7=1;
else maxRtP7=0;
 end
if interMax7a > maxRange7
    maxRfN7=1;
else maxRfN7=0;
 end
if interMin7a < minRange7
    minRtP7=1;
else minRtP7=0;

end

if interMin7a > minRange7
    minRfN7=1;
```

```
else minRfN7=0;
  end

if interMax7b < intraMax7
    maxfP7=1 ;
else maxfP7=0;
end
if interMax7b > intraMax7
    maxtN7=1;
else maxtN7=0;
end

if interAvg7b < intraAvg7
    avgfP7=1;
else  avgfP7=0;
end
if  interAvg7b > intraAvg7
    avgtN7=1;
else avgtN7=0;
end
if interMin7b < intraMin7
    minfP7=1;
else minfP7=0;
end
if interMin7b > intraMin7
    mintN7=1;
else mintN7=0;
end

if interMax7b < maxRange7
    maxRfP7=1;
else maxRfP7=0;

end
```

```
if interMax7b > maxRange7
    maxRtN7=1;
else maxRtN7=0;
  end


if interMin7b < minRange7
    minRfP7=1;
else minRfP7=0;

end
if interMin7b > minRange7
    minRtN7=1;
else minRtN7=0;
  end
if interMax8a < intraMax8
    maxtP8=1 ;
else maxtP8=0;
end

if interMax8a > intraMax8
    maxfN8=1;
else maxfN8=0;
end

if interAvg8a < intraAvg8
    avgtP8=1;
else  avgtP8=0;
end
if  interAvg8a > intraAvg8
    avgfN8=1;
else avgfN8=0;
end
if interMin8a < intraMin8
```

```
    mintP8=1;
else mintP8=0;
end

if interMin8a > intraMin8
    minfN8=1;
else minfN8=0;
end

if interMax8a < maxRange8
    maxRtP8=1;
else maxRtP8=0;
 end
if interMax8a > maxRange8
    maxRfN8=1;
else maxRfN8=0;
end
if interMin8a < minRange8
    minRtP8=1;
else minRtP8=0;
 end
if interMin8a > minRange8
    minRfN8=1;
else minRfN8=0;

end
if interMax8b < intraMax8
    maxfP8=1 ;
else maxfP8=0;
end

if interMax8b > intraMax8
    maxtN8=1;
else maxtN8=0;
```

```
end

if interAvg8b < intraAvg8
    avgfP8=1;
else  avgfP8=0;
end
if  interAvg8b > intraAvg8
    avgtN8=1;
else avgtN8=0;
end
if interMin8b < intraMin8
    minfP8=1;
else minfP8=0;
end

if interMin8b > intraMin8
    mintN8=1;
else mintN8=0;
end

if interMax8b < maxRange8
    maxRfP8=1;
else maxRfP8=0;

end
if interMax8b > maxRange8
    maxRtN8=1;
else maxRtN8=0;
end

if interMin8b < minRange8
    minRfP8=1;
else minRfP8=0;
```

```
end

if interMin8b > minRange8
    minRtN8=1;
else minRtN8=0;

end
if interMax9a < intraMax9
    maxtP9=1 ;
else maxtP9=0;
end
if interMax9a > intraMax9
    maxfN9=1;
else maxfN9=0;
end

if interAvg9a < intraAvg9
    avgtP9=1;
else  avgtP9=0;
end
if  interAvg9a > intraAvg9
    avgfN9=1;
else avgfN9=0;
end
if interMin9a < intraMin9
    mintP9=1;
else mintP9=0;
end

if interMin9a > intraMin9
    minfN9=1;
else minfN9=0;
end
if interMax9a < maxRange9
```

```
    maxRtP9=1;
else maxRtP9=0;
end

if interMax9a > maxRange9
    maxRfN9=1;
else maxRfN9=0;

end

if interMin9a < minRange9
    minRtP9=1;
else minRtP9=0;
 end
if interMin9a > minRange9
    minRfN9=1;
else minRfN9=0;

end

if interMax9b < intraMax9
    maxfP9=1 ;
else maxfP9=0;
end

if interMax9b > intraMax9
    maxtN9=1;
else maxtN9=0;
end
if interAvg9b < intraAvg9
    avgfP9=1;
else  avgfP9=0;
end
if  interAvg9b > intraAvg9
```

```
    avgtN9=1;
else avgtN9=0;
end
if interMin9b < intraMin9
    minfP9=1;
else minfP9=0;
end
if interMin9b > intraMin9
    mintN9=1;
else mintN9=0;
end

if interMax9b < maxRange9
    maxRfP9=1;
else maxRfP9=0;

end
if interMax9b > maxRange9
    maxRtN9=1;
else maxRtN9=0;

end



if interMin9b < minRange9
    minRfP9=1;
else minRfP9=0;

end
if interMin9b > minRange9
    minRtN9=1;
else minRtN9=0;
 end
if interMax10a < intraMax10
```

```
    maxtP10=1 ;
else maxtP6=0;
end
if interMax10a > intraMax10
    maxfN10=1;
else maxfN10=0;
end

if interAvg10a < intraAvg10
    avgtP10=1;
else  avgtP10=0;
end
if  interAvg10a > intraAvg10
    avgfN10=1;
else avgfN10=0;
end
if interMin10a < intraMin10
    mintP10=1;
else mintP10=0;
end

if interMin10a > intraMin10
    minfN10=1;
else minfN10=0;
end

if interMax10a < maxRange10
    maxRtP10=1;
else maxRtP10=0;

end
if interMax10a > maxRange10
    maxRfN10=1;
else maxRfN10=0;
```

```
end

if interMin10a < minRange10
    minRtP10=1;
else minRtP10=0;

end
if interMin10a > minRange10
    minRfN10=1;
else minRfN10=0;
  end

if interMax10b < intraMax10
    maxfP10=1 ;
else maxfP10=0;
end

if interMax10b > intraMax10
    maxtN10=1;
else maxtN10=0;
end

if interAvg10b < intraAvg10
    avgfP10=1;
else  avgfP10=0;
end
if  interAvg10b > intraAvg10
    avgtN10=1;
else avgtN10=0;
end
if interMin10b < intraMin10
    minfP10=1;
else minfP10=0;
```

```
end
if interMin10b > intraMin10
    mintN10=1;
else mintN10=0;
end
if interMax10b < maxRange10
    maxRfP10=1;
else maxRfP10=0;

end

if interMax10b > maxRange10
    maxRtN10=1;
else maxRtN10=0;

end
if interMin10b < minRange10
    minRfP10=1;
else minRfP10=0;

end

if interMin10b > minRange10
    minRtN10=1;
else minRtN10=0;

end
if interMax11a < intraMax11
    maxtP11=1 ;
else maxtP11=0;
end
if interMax11a > intraMax11
    maxfN11=1;
else maxfN11=0;
```

```
end

if interAvg11a < intraAvg11
    avgtP11=1;
else  avgtP11=0;
end
if  interAvg11a > intraAvg11
    avgfN11=1;
else avgfN11=0;
end
if interMin11a < intraMin11
    mintP11=1;
else mintP11=0;
end

if interMin11a > intraMin11
    minfN11=1;
else minfN11=0;
end
if interMax11a < maxRange11
    maxRtP11=1;
else maxRtP11=0;

end

if interMax11a > maxRange11
    maxRfN11=1;
else maxRfN11=0;

end

if interMin11a < minRange11
    minRtP11=1;
else minRtP11=0;
```

```
end
if interMin11a > minRange11
    minRfN11=1;
else minRfN11=0;

end

if interMax11b < intraMax11
    maxfP11=1 ;
else maxfP11=0;
end

if interMax11b > intraMax11
    maxtN11=1;
else maxtN11=0;
end
if interAvg11b < intraAvg11
    avgfP11=1;
else  avgfP11=0;
end
if  interAvg11b > intraAvg11
    avgtN11=1;
else avgtN11=0;
end
if interMin11b < intraMin11
    minfP11=1;
else minfP11=0;
end
if interMin11b > intraMin11
    mintN11=1;
else mintN11=0;
end
```

```
if interMax11b < maxRange11
    maxRfP11=1;
else maxRfP11=0;

end
if interMax11b > maxRange11
    maxRtN11=1;
else maxRtN11=0;

end



if interMin11b < minRange11
    minRfP11=1;
else minRfP11=0;

end
if interMin11b > minRange11
    minRtN11=1;
else minRtN11=0;

end
if interMax12a < intraMax12
    maxtP12=1 ;
else maxtP12=0;
end
if interMax12a > intraMax12
    maxfN12=1;
else maxfN12=0;
end

if interAvg12a < intraAvg12
    avgtP12=1;
else  avgtP12=0;
```

```
end
if  interAvg12a > intraAvg12
    avgfN12=1;
else avgfN12=0;
end
if interMin12a < intraMin12
    mintP12=1;
else mintP12=0;
end

if interMin12a > intraMin12
    minfN12=1;
else minfN12=0;
end

if interMax12a < maxRange12
    maxRtP12=1;
else maxRtP12=0;

end
if interMax12a > maxRange12
    maxRfN12=1;
else maxRfN12=0;

end

if interMin12a < minRange12
    minRtP12=1;
else minRtP12=0;

end
if interMin12a > minRange12
    minRfN12=1;
else minRfN12=0;
```

```
end

if interMax12b < intraMax12
    maxfP12=1 ;
else maxfP12=0;
end

if interMax12b > intraMax12
    maxtN12=1;
else maxtN12=0;
end
if interAvg12b < intraAvg12
    avgfP12=1;
else  avgfP12=0;
end
if  interAvg12b > intraAvg12
    avgtN12=1;
else avgtN12=0;
end
if interMin12b < intraMin12
    minfP12=1;
else minfP12=0;
end

if interMin12b > intraMin12
    mintN12=1;
else mintN12=0;
end
if interMax12b < maxRange12
    maxRfP12=1;
else maxRfP12=0;

end
```

```
if interMax12b > maxRange12
    maxRtN12=1;
else maxRtN12=0;

end

if interMax12b < minRange12
    minRfP12=1;
else minRfP12=0;
end
if interMax12b > minRange12
    minRtN12=1;
else minRtN12=0;

end
if interMax13a < intraMax13
    maxtP13=1 ;
else maxtP13=0;
end
if interMax13a > intraMax13
    maxfN13=1;
else maxfN13=0;
end
if interAvg13a < intraAvg13
    avgtP13=1;
else  avgtP13=0;
end
if  interAvg13a > intraAvg13
    avgfN13=1;
else avgfN13=0;
end
if interMin13a < intraMin13
    mintP13=1;
```

```
else mintP13=0;
end

if interMin13a > intraMin13
    minfN13=1;
else minfN13=0;
end

if interMax13a < maxRange13
    maxRtP13=1;
else maxRtP13=0;

end

if interMax13a > maxRange13
    maxRfN13=1;
else maxRfN13=0;

end

if interMin13a < minRange13
    minRtP13=1;
else minRtP13=0;

end

if interMin13a > minRange13
    minRfN13=1;
else minRfN13=0;

end
if interMax13b < intraMax13
    maxfP13=1 ;
else maxfP13=0;
```

```
end

if interMax13b > intraMax13
    maxtN13=1;
else maxtN13=0;
end

if interAvg13b < intraAvg13
    avgfP13=1;
else  avgfP13=0;
end
if  interAvg13b > intraAvg13
    avgtN13=1;
else avgtN13=0;
end
if interMin13b < intraMin13
    minfP13=1;
else minfP13=0;
end

if interMin13b > intraMin13
    mintN13=1;
else mintN13=0;
end

if interMax13b < maxRange13
    maxRfP13=1;
else maxRfP13=0;

end
if interMax13b > maxRange13
    maxRtN13=1;
else maxRtN13=0;
```

```
end



if interMin13b < minRange13
    minRfP13=1;
else minRfP13=0;

end

if interMin13b > minRange13
    minRtN13=1;
else minRtN13=0;

end
if interMax14a < intraMax14
    maxtP14=1 ;
else maxtP14=0;
end
if interMax14a > intraMax14
    maxfN14=1;
else maxfN14=0;
end
if interAvg14a < intraAvg14
    avgtP14=1;
else  avgtP14=0;
end
if  interAvg14a > intraAvg14
    avgfN14=1;
else avgfN14=0;
end
if interMin14a < intraMin14
    mintP14=1;
else mintP14=0;
end
```

```
if interMin14a > intraMin14
    minfN14=1;
else minfN14=0;
end

if interMax14a < maxRange14
    maxRtP14=1;
else maxRtP14=0;

end

if interMax14a > maxRange14
    maxRfN14=1;
else maxRfN14=0;
 end
if interMin14a < minRange14
    minRtP14=1;
else minRtP14=0;

end

if interMin14a > minRange14
    minRfN14=1;
else minRfN14=0;
 end

if interMax14b < intraMax14
    maxfP14=1 ;
else maxfP14=0;
end

if interMax14b > intraMax14
    maxtN14=1;
else maxtN14=0;
```

```
end

if interAvg14b < intraAvg14
    avgfP14=1;
else  avgfP14=0;
end
if  interAvg14b > intraAvg14
    avgtN14=1;
else avgtN14=0;
end
if interMin14b < intraMin14
    minfP14=1;
else minfP14=0;
end

if interMin14b > intraMin14
    mintN14=1;
else mintN14=0;
end

if interMax14b < maxRange14
    maxRfP14=1;
else maxRfP14=0;

end

if interMax14b > maxRange14
    maxRtN14=1;
else maxRtN14=0;

end



if interMin14b < minRange14
```

```
    minRfP14=1;
else minRfP14=0;

end

if interMin14b > minRange14
    minRtN14=1;
else minRtN14=0;

end
if interMax15a < intraMax15
    maxtP15=1 ;
else maxtP15=0;
end

if interMax15a > intraMax15
    maxfN15=1;
else maxfN15=0;
end

if interAvg15a < intraAvg15
    avgtP15=1;
else  avgtP15=0;
end
if  interAvg15a > intraAvg15
    avgfN15=1;
else avgfN15=0;
end
if interMin15a < intraMin15
    mintP15=1;
else mintP15=0;
end

if interMin15a > intraMin15
```

```
    minfN15=1;
else minfN15=0;
end
if interMax15a < maxRange15
    maxRtP15=1;
else maxRtP15=0;
 end
if interMax15a > maxRange15
    maxRfN15=1;
else maxRfN15=0;
end
if interMin15a < minRange15
    minRtP15=1;
else minRtP15=0;

end

if interMin15a > minRange15
    minRfN15=1;
else minRfN15=0;

end

if interMax15b < intraMax15
    maxfP15=1 ;
else maxfP15=0;
end
if interMax15b > intraMax15
    maxtN15=1;
else maxtN15=0;
end
if interAvg15b < intraAvg15
    avgfP15=1;
else  avgfP15=0;
```

```
end
if  interAvg15b > intraAvg15
    avgtN15=1;
else avgtN15=0;
end
if interMin15b < intraMin15
    minfP15=1;
else minfP15=0;
end

if interMin15b > intraMin15
    mintN15=1;
else mintN15=0;
end

if interMax15b < maxRange15
    maxRfP15=1;
else maxRfP15=0;
  end

if interMax15b > maxRange15
    maxRtN15=1;
else maxRtN15=0;

end


if interMin15b < minRange15
    minRfP15=1;
else minRfP15=0;
 end
if interMin15b > minRange15
    minRtN15=1;
else minRtN15=0;
```

```
end
if interMax16a < intraMax16
    maxtP16=1 ;
else maxtP16=0;
end

if interMax16a > intraMax16
    maxfN16=1;
else maxfN16=0;
end
if interAvg16a < intraAvg16
    avgtP16=1;
else  avgtP16=0;
end
if  interAvg16a > intraAvg16
    avgfN16=1;
else avgfN16=0;
end
if interMin16a < intraMin16
    mintP16=1;
else mintP16=0;
end

if interMin16a > intraMin16
    minfN16=1;
else minfN16=0;
end
if interMax16a < maxRange16
    maxRtP16=1;
else maxRtP16=0;

end
```

```
if interMax16a > maxRange16
    maxRfN16=1;
else maxRfN16=0;
end
if interMin16a < minRange16
    minRtP16=1;
else minRtP16=0;

end

if interMin16a > minRange16
    minRfN16=1;
else minRfN16=0;

end
if interMax16b < intraMax16
    maxfP16=1 ;
else maxfP16=0;
end

if interMax16b > intraMax16
    maxtN16=1;
else maxtN16=0;
end

if interAvg16b < intraAvg16
    avgfP16=1;
else  avgfP16=0;
end
if  interAvg16b > intraAvg16
    avgtN16=1;
else avgtN16=0;
end
if interMin16b < intraMin16
```

```
    minfP16=1;
else minfP16=0;
end

if interMin16b > intraMin16
    mintN16=1;
else mintN16=0;
end

if interMax16b < maxRange16
    maxRfP16=1;
else maxRfP16=0;

end

if interMax16b > maxRange16
    maxRtN16=1;
else maxRtN16=0;
 end
if interMin16b < minRange16
    minRfP16=1;
else minRfP16=0;
end
if interMin16b > minRange16
    minRtN16=1;
else minRtN16=0;

end
if interMax17a < intraMax17
    maxtP17=1 ;
else maxtP17=0;
end
if interMax17a > intraMax17
    maxfN17=1;
```

```
else maxfN17=0;
end
if interAvg17a < intraAvg17
    avgtP17=1;
else  avgtP17=0;
end
if  interAvg17a > intraAvg17
    avgfN17=1;
else avgfN17=0;
end
if interMin17a < intraMin17
    mintP17=1;
else mintP17=0;
end

if interMin17a > intraMin17
    minfN17=1;
else minfN17=0;
end

if interMax17a < maxRange17
    maxRtP17=1;
else maxRtP17=0;

end
if interMax17a > maxRange17
    maxRfN17=1;
else maxRfN17=0;

end

if interMin17a < minRange17
    minRtP17=1;
else minRtP17=0;
```

```
end
if interMin17a > minRange17
    minRfN17=1;
else minRfN17=0;

end

if interMax17b < intraMax17
    maxfP17=1 ;
else maxfP17=0;
end
if interMax17b > intraMax17
    maxtN17=1;
else maxtN17=0;
end
if interAvg17b < intraAvg17
    avgfP17=1;
else  avgfP17=0;
end
if  interAvg17b > intraAvg17
    avgtN17=1;
else avgtN17=0;
end
if interMin17b < intraMin17
    minfP17=1;
else minfP17=0;
end

if interMin17b > intraMin17
    mintN17=1;
else mintN17=0;
end
if interMax17b < maxRange17
```

```
    maxRfP17=1;
else maxRfP17=0;

end
if interMax17b > maxRange17
    maxRtN17=1;
else maxRtN17=0;

end

if interMin17b < minRange17
    minRfP17=1;
else minRfP17=0;

end

if interMin17b > minRange17
    minRtN17=1;
else minRtN17=0;
 end
if interMax18a < intraMax18
    maxtP18=1 ;
else maxtP18=0;
end
if interMax18a > intraMax18
    maxfN18=1;
else maxfN18=0;
end

if interAvg18a < intraAvg18
    avgtP18=1;
else  avgtP18=0;
end
if  interAvg18a > intraAvg18
```

```
    avgfN18=1;
else avgfN18=0;
end
if interMin18a < intraMin18
    mintP18=1;
else mintP18=0;
end

if interMin18a > intraMin18
    minfN18=1;
else minfN18=0;
end
if interMax18a < maxRange18
    maxRtP18=1;
else maxRtP18=0;

end

if interMax18a > maxRange18
    maxRfN18=1;
else maxRfN18=0;

end
if interMin18a < minRange18
    minRtP18=1;
else minRtP18=0;

end

if interMin18a > minRange18
    minRfN18=1;
else minRfN18=0;

end
```

```
if interMax18b < intraMax18
    maxfP18=1 ;
else maxfP18=0;
end
if interMax18b > intraMax18
    maxtN18=1;
else maxtN18=0;
end

if interAvg18b < intraAvg18
    avgfP18=1;
else  avgfP18=0;
end
if  interAvg18b > intraAvg18
    avgtN18=1;
else avgtN18=0;
end
if interMin18b < intraMin18
    minfP18=1;
else minfP18=0;
end

if interMin18b > intraMin18
    mintN18=1;
else mintN18=0;
end

if interMax18b < maxRange18
    maxRfP18=1;
else maxRfP18=0;
 end

if interMax18b > maxRange18
```

```
    maxRtN18=1;
else maxRtN18=0;

end
if interMin18b < minRange18
    minRfP18=1;
else minRfP18=0;

end

if interMin18b > minRange18
    minRtN18=1;
else minRtN18=0;
 end

maxtp=[maxtP1 maxtP2 maxtP3 maxtP4 maxtP5 maxtP6 maxtP7
maxtP8 maxtP9 maxtP10 maxtP11 maxtP12 maxtP13
maxtP14 maxtP15 maxtP16 maxtP17 maxtP18];
MaxTP=sum(maxtp)
maxfn=[maxfN1 maxfN2 maxfN3 maxfN4 maxfN5 maxfN6
 maxfN7 maxfN8 maxfN9 maxfN10 maxfN11 maxfN12
maxfN13 maxfN14 maxfN15 maxfN16  maxfN17 maxfN18];
MaxFN=sum(maxfn)
avgtp=[avgtP1 avgtP2 avgtP3 avgtP4 avgtP5 avgtP6
 avgtP7 avgtP8 avgtP9 avgtP10 avgtP11 avgtP12 avgtP13
avgtP14 avgtP15 avgtP16 avgtP17 avgtP18];
AvgTP=sum(avgtp)
avgfn= [avgfN1 avgfN2 avgfN3 avgfN4 avgfN5 avgfN6
 avgfN7 avgfN8 avgfN9 avgfN10 avgfN11 avgfN12
 avgfN13 avgfN14 avgfN15 avgfN16 avgfN17 avgfN18];
AvgFN=sum(avgfn)
mintp=[mintP1 mintP2 mintP3 mintP4 mintP5 mintP6
mintP7 mintP8 mintP9 mintP10 mintP11 mintP12
mintP13 mintP14 mintP15 mintP16 mintP17 mintP18];
```

```
MinTP=sum(mintp)
minfn=[minfN1 minfN2 minfN3 minfN4 minfN5 minfN6
minfN7 minfN8 minfN9 minfN10 minfN11 minfN12
minfN13 minfN14 minfN15 minfN16 minfN17 minfN18];
MinFN=sum(minfn)
maxrtp=[maxRtP1 maxRtP2 maxRtP3 maxRtP4 maxRtP5
maxRtP6 maxRtP7 maxRtP8 maxRtP9 maxRtP10 maxRtP11 maxRtP12
 maxRtP13 maxRtP14 maxRtP15 maxRtP16 maxRtP17 maxRtP18];
MaxRTP=sum(maxrtp)
maxrfn= [maxRfN1 maxRfN2 maxRfN3 maxRfN4 maxRfN5
maxRfN6 maxRfN7 maxRfN8 maxRfN9 maxRfN10 maxRfN11 maxRfN12
 maxRfN13 maxRfN14 maxRfN15 maxRfN16 maxRfN17 maxRfN18];
MaxRFN=sum(maxrfn)
minrtp=[minRtP1 minRtP2 minRtP3 minRtP4 minRtP5 minRtP6
minRtP7 minRtP8 minRtP9 minRtP10 minRtP11 minRtP12
 minRtP13 minRtP14 minRtP15 minRtP16 minRtP17 minRtP18];
MinRTP=sum(minrtp)
minrfn=[minRfN1 minRfN2 minRfN3 minRfN4 minRfN5
 minRfN6 minRfN7 minRfN8 minRfN9 minRfN10 minRfN11
minRfN12 minRfN13 minRfN14 minRfN15 minRfN16 minRfN17 minRfN18];
MinRFN=sum(minrfn)
maxfp=[maxfP1 maxfP2 maxfP3 maxfP4 maxfP5 maxfP6
maxfP7 maxfP8 maxfP9 maxfP10 maxfP11 maxfP12 maxfP13
maxfP14 maxfP15 maxfP16 maxfP17 maxfP18];
MaxFP=sum(maxfp)
maxtn=[maxtN1 maxtN2 maxtN3 maxtN4 maxtN5 maxtN6 maxtN7
 maxtN8 maxtN9 maxtN10 maxtN11 maxtN12 maxtN13
maxtN14 maxtN15 maxtN16 maxtN17 maxtN18];
MaxTN=sum(maxtn)
avgfp=[avgfP1 avgfP2 avgfP3 avgfP4 avgfP5
avgfP6 avgfP7 avgfP8 avgfP9 avgfP10 avgfP11 avgfP12
avgfP13 avgfP14 avgfP15 avgfP16 avgfP17 avgfP18];
AvgFP=sum(avgfp)
avgtn= [avgtN1 avgtN2 avgtN3 avgtN4 avgtN5 avgtN6
```

```
 avgtN7 avgtN8 avgtN9 avgtN10 avgtN11 avgtN12
avgtN13 avgtN14 avgtN15 avgtN16 avgtN17 avgtN18];
AvgTN=sum(avgtn)
minfp=[minfP1 minfP2 minfP3 minfP4 minfP5 minfP6
minfP7 minfP8 minfP9 minfP10 minfP11 minfP12
minfP13 minfP14 minfP15 minfP16 minfP17 minfP18];
MinFP=sum(minfp)
mintn=[mintN1 mintN2 mintN3 mintN4 mintN5 mintN6
mintN7 mintN8 mintN9 mintN10 mintN11 mintN12
mintN13 mintN14 mintN15 mintN16 mintN17 mintN18];
MinTN=sum(mintn)
maxrfp=[maxRfP1 maxRfP2 maxRfP3 maxRfP4 maxRfP5
 maxRfP6 maxRfP7 maxRfP8 maxRfP9 maxRfP10 maxRfP11
maxRfP12 maxRfP13 maxRfP14 maxRfP15 maxRfP16 maxRfP17 maxRfP18];
MaxRFP=sum(maxrfp)
maxrtn= [maxRtN1 maxRtN2 maxRtN3 maxRtN4 maxRtN5
 maxRtN6 maxRtN7 maxRtN8 maxRtN9 maxRtN10 maxRtN11
maxRtN12 maxRtN13 maxRtN14 maxRtN15 maxRtN16 maxRtN17 maxRtN18];
MaxRTN=sum(maxrtn)
minrfp=[minRfP1 minRfP2 minRfP3 minRfP4 minRfP5
minRfP6 minRfP7 minRfP8 minRfP9 minRfP10 minRfP11
 minRfP12 minRfP13 minRfP14 minRfP15 minRfP16 minRfP17 minRfP18];
MinRFP=sum(minrfp)
minrtn=[minRtN1 minRtN2 minRtN3 minRtN4 minRtN5
minRtN6 minRtN7 minRtN8 minRtN9 minRtN10 minRtN11
 minRtN12 minRtN13 minRtN14 minRtN15 minRtN16 minRtN17 minRtN18];
MinRTN=sum(minrtn)
sensitivityUSINGMAX=MaxTP/(MaxTP+MaxFN)*100
specificityUSINGMAX=MaxTN/(MaxTN+MaxFP)*100
sensitivityUSINGAVG=AvgTP/(AvgTP+AvgFN)*100
specificityUSINGAVG=AvgTN/(AvgTN+AvgFP)*100
sensitivityUSINGMIN=MinTP/(MinTP+MinFN)*100
specificityUSINGMIN=MinTN/(MinTN+MinFP)*100
sensitivityUSINGMAXRANGE=MaxRTP/(MaxRTP+MaxRFN)*100
```

```
specificityUSINGMAXRANGE=MaxRTN/(MaxRTN+MinRFP)*100
sensitivityUSINGMINRANGE=MinRTP/(MinRTP+MinRFN)*100
specificityUSINGMINRANGE=MinRTN/(MinRTN+MinRFP)*100
```

6.2 Appendix B

Figure 6.1 shows signatures used in this project.

Figure 6.1: Signatures used in the project.

Bibliography

[1] B. Herbst. J. Coetzer. and J. Preez, "Online Signature Verification Using the Discrete Radon Transform and a Hidden Markov Model," *EURASIP.Journal on Applied Signal Processing*, vol. 4, pp. 559–571, 2004.

[2] D. Lowe, "Distinctive Image features from Scale- invariant Keypoints.," *International Journal of Computer Vision.*, vol. 60, no. 2, pp. 91–110, 2004.

[3] R. Plamondon and S. N. Srihari, " On-line and Off-line handwriting recognition," *IEEE Trans.on Pattern Analysis and machine Intelligence*, vol. 22, no. 1, pp. 63–84, 2000.

[4] S. I. Abuhaiba, " Offline Signature Verification Using Graph Matching," *Turk J Elec Engine*, vol. 15, no. 1, 2007.

[5] A. I. Abdullah, " Handwritten Signature Verification Using Image Invariants and Dynamic Features," *Proceedings of the International Conference on Computer Graphics, Imaging and Visualisation*, 2006.

[6] G. F. Russel. A. Heilper. B. A. Smith. J. Hu. D.Markman. J. E. Graham. T. G. Zimmerman. and C. Drews, " Retail Application of Signature Verification," *Proceedings of SPIE 2004*, vol. 5404, pp. 206–214, August 2004.

[7] S. Srihari. K. M. Kalera. and A. XU, "Offline Signature Verification and Identification Using Distance Statistics," *International Journal of Pattern Recognition And Artificial Intelligence*, vol. 18, no. 7, pp. 1339–1360, 2004.

[8] S. Reddy. B. Maghi. and P. Babu, "Novel Features for Offline signature verification.," *Journal of Computer,Communication and Control.*, vol. 1, pp. 17–24, 2006.

[9] B. A. Jesus. A. Migual. and M. Traveiso, " Off-line Geometric Parameters for Automatic Signature Verification Using Fixed Point Arithemetic," *IEEE Trans.Pattern Analysis and Machine Intelligence*, vol. 27, no. 6, pp. 341–356, June 2005.

[10] K. B. Viyanak, "A color code Algorithm for Signature Recognition," *International Journal of Pattern Recognition And Artificial Intelligence*, vol. 6, no. 1, pp. 1–12, 2007.

[11] Z. Lin. W. Liang. and R. C. Zhao, "Offline signature verification Incorporating the prior model," *International Conference on Machine Learning and Cybernetics*, vol. 3, pp. 1602–1606, 2003.

[12] T. Ṣenturk. E. Özgunduz. and E. Karshgil, " Handwritten Signature Verification Using Image Invariants and Dynamic Features," *Proceedings of the 13th European Signal Processing Conference EUSIPCO 2005,Antalya Turkey*, 4th-8th September, 2005.

[13] B. C. Lovell. V. K. Madasu. and K. Kubik, "Automatic Handwritten Signature verification system for Australian Passports," *Science,Engineering and Technology Summit on Counter-Terrorism,Canberra*, pp. 53–66, 2004.

[14] H. S. Srihari and M. Beall, "Signature Verifcation Using Kolmogrov Smirnov Statistic," *Proceedings of International Graphonomics Society,Salemo Italy*, pp. 152–156, june,2005.

[15] Check fraud statistics, "National fraud centre," http://www.ckfraud.org/statistics.html - Retrieved february 22,2008, 2008.

[16] Embassy of the United States Kampala Uganda, "Business fraud warning," http://kampala.usembassy.gov/business fraud warning2.html - Retrieved february 22,2008, 2008.

[17] Bank of Uganda, "Bankfraud," http://www.bou.or.ug/BANKFRAUD.pdf-Retrieved february 22,2008, 2008.

[18] S. N. Srihari and A. Xu., " Learning Strategies and Classification Methods for Offline Signature Verification," *Proceedings of the 7th international Workshop on Frontiers in handwriting recognition*, 2004.

[19] F. Bortolozi. E. R. Justino., A. E. Yocoubi. and R. Sabourin, "An Off-line Signature Verification System Using HMM and Graphometric features," *DAS 2000,4th IAPR International on Document Analysis Systems,Rio de Jeneiro*, 2000.

[20] K. Faez. M. Dehghan. and M. Fathi, "Signature Verification Using Shape Descriptor and Multiple Neural Network," *IEEE TENCON 1997-Speech and Image Technologies For Computing and Telecomunications*, pp. 415–418, 1997.

[21] H. Hammandlu and V. M. Krishna, " Off-line Signature Verification and Forgery detection using Fuzzy modeling," *Pattern Recognition*, vol. 38, pp. 341–356, 2005.

[22] M. Blumenstein. S. Armand. and Muthukkumarasamy, "Off-line Signature Verification using the Enhanced Modified Direction Feature and Neuralbased Classification," *International Joint Conference on Neural Networks*, 2006.

[23] Q. Qianghua. S. Yaiqian and P. Jingui, "Offline Signature Verification Using Geometric Features Specific to Chinese Handwritting," *24th Int. Conf.Information Technology Interfaces*, June 24-27,2002.

[24] Y. Y. Wang. C. H. Leung. Y. Y. Tang. P. C. K. Kwok. K. W. E. Tse. B. Fang and Y. K. Wong, "A Smoothness Index Based Approach for Off-line Signature Verification," *Proceedings of the Fifth International Conference on Document Analysis and Recognition*, pp. 785–787, September 9,1999.

[25] Y. Y. wang. B. Fang. and C. H. Leung, "Offline Signature verification by analysis of cursive stroke," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 15, no. 4, pp. 659–673, 2001.

[26] H. Miike. Y. Mizukami., M. Yoshimura and I. Yoshimura, "An Offline signature verification system using extracted displacement function," *Pattern Recognition Letter*, vol. 23, no. 13, pp. 1569–1577, 2002.

[27] C. H. Leung. Y. Y. Tang. P. C. K. Kwok. K. W. Tse. B. Fang. and Y. K. Wong, "Off-line signature verification with generated Training samples.," *IEEE proceedings Vision,Image and Signal processing.*, vol. 149, no. 2, pp. 85–90, 2002.

[28] D. Lowe, "Object Recognition from Local Scale Invariant features.," *In International Conference on Computer Vision*, pp. 1150–1157, 1999.

[29] H. Kim. H. Lee. and H. K. Lee, "Robust Image Watermarking using Local Invariant Features," *Proceedings of SPIE*, vol. 45, no. 3, 2006.

[30] G. Enrico. B. Manuele., L. Anderea. and T. Massimo, "On the use of SIFT features for face authentication.," *In the proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop*, pp. 91–110, 2006.

[31] P. Schwarz, "Recognition of Graffiti," *BS Thesis,The University of Western Australia*, 2006.

[32] L. Dlagnekov, "Video-based Car Surveillance: Licence plate, Make and Model Recognition," *MSc Thesis,University of Calfornia,San Diego*, 2005.

[33] P. Sharath. P. UnSang. and A. K. Jain, "Robust Image Watermarking using Local Invariant Features," *Proceedings of SPIE Defense and Security symposium Orlando,Florida*, 2008.

[34] T.F. EL-Maraghi, "Matlab sift tutorial," Available from: ftp://ftp.cs.utoronto.ca/pub/jepson/teaching/vision/2503/SIFTtutorial.zip - Retrieved July 5,2008.

[35] I. H. Witten and E. Franh, *Data Mining*, Elesevier, 2005.

[36] B. Zhang. S. N. Srihari., C. I. Tomai. and S. J. Lee, "Individuality of Numerals.," *Proceedings International Conference on Document Analysis and Recognition (ICDAR) Edinburgh, Scotland*, pp. 1096–1100, 2003.

[37] D. de Ridder. F. van der Heijden., R. P. W. Duinn. and D. M. J. Tax, *Classification,Parameter Estimation and State Estimation:An Engineering Approach using MATLAB*, John Wiley and Sons Ltd, 2004.