DECEMBER 2017 M.Sc. in Electrical and Electronics Engineering ABDULBARI TALIB NASER AL AZZAWI

UNIVERSITY OF GAZIANTEP

GRADUATE SCHOOL OF

NATURAL & APPLIED SCIENCES

SIGNATURE RECOGNITION BY USING SIFT AND SURF WITH SVM BASIC ON RBF FOR VOTING ONLINE

M. Sc. THESIS

IN

ELECTRICAL AND ELECTRONICS ENGINEERING

BY

ABDULBARI TALIB NASER AL AZZAWI

DECEMBER 2017

Signature Recognition by Using SIFT and SURF with SVM Basic on RBF for Voting Online

M.Sc. Thesis

in

Electrical and Electronics Engineering

University of Gaziantep

Supervisor

Prof. Dr. NURAN DOGRU

by

Abdulbari Talib Naser AL AZZAWI

December 2017

REPUBLIC OF TURKEY
UNIVERSITY OF GAZIANTEP
GRADUATE SCHOOL OF NATURAL & APPLIED SCIENCES
ELECTRICAL AND ELECTRONICS ENGINEERING

Name of the thesis: Signature recognition by using SIFT and SURF with SVM basic on RBF for voting online

Name of the student: Abdulbari Talib Naser AL AZZAWI

Exam Date: 15/12/2017

Approval of Graduate School of Natural and Applied Sciences

[illegible]

Prof. Dr. Ahmet Necmeddin YAZICI

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

E. Erçelebi

Prof. Dr. Ergun ERÇELEBİ

Head of Department

This is to certify that we have read this thesis and that in our consensus opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Prof. Dr. Nuran DOGRU [illegible]

Supervisor

Examining Committee Members	Signature
Prof. Dr. Nuran DOGRU	[illegible]
Assoc. Prof. Dr. Kemal DELIHACIOGLU	[illegible]
Assoc. Prof. Dr. Sema KAYHAN	[illegible]

ABSTRACT

SIGNATURE RECOGNITION BY USING SIFT AND SURF WITH SVM BASIC ON RBF FOR VOTIN ONLINE

AL-AZZAWI, Abdulbari Talib Naser

M.Sc. in Electrical and Electronics Engineering

Supervisor: Prof. Dr. NURAN DOGRU

December 2017

111 page

The Signature recognition is known as the process to verify a writer by examining the signature upon samples has been studied and stored in the database .This process has two types: The offline and the online. This thesis deals with the offline technique and proposed a SIFT and a SURF algorithm which is used to detector and descriptor keypoint (features) for each signature image. This process, Bag-of-word features, is operated by making vector quantization technique, which is outlined the key points for each training image inside a unified dimensional histogram. Features of bag-of-word are put inside multiclass Support Vector Machine (SVM) classifier established upon the Radial Basis Function (RBF) for a training and testing. Open CV C++ is used as an image processing tool and tool for feature extraction. In this thesis, the performance of SIFT on SVM based RBF kernel is compared with SURF on SVM based RBF kernel .It is found that the use of SIFT with SVM-RBF kernel system, has an accuracy of 98.75% and SURF with SVM-RBF kernel has an accuracy of 97.5%.We used TCP/IP that uses the client/server model of communication in which a computer user requests and is provided a service by another computer in the network. The client server network communication system is implemented by using java programming language for application interface and use to detect the network feature and use java serializable interface method to store the image of the system and important data of the system.

Key Words: Signature recognition, SIFT, SURF, SVM-RBF kernel, BOW (bag of words), Open CV C++, TCP/IP.

ÖZET

ONLİNE OYLAMA İÇİN RBF'DE SVM TEMELLİ SIFT VE SURF KULLANARAK İMZA TANIMASI

AL-AZZAWI, Abdulbari Talib Naser
Yüksek Lisans Tezi, Elektrik-Elektronik Müh. Bölümü
Tez Yöneticisi: Prof. Dr. Nuran Dogru
Aralık 2017
111 sayfa

İmza tanıma, bir yazarın örnek üzerinde incelenerek doğruluğunu incelemek ve veritabanında depolanması işlemidir. Bu işlemin iki türü vardır: çevrimdışı ve çevrimiçi. Bu tez, offline tekniği ele alıyor ve arttırılmış çevrimdışı imza tanıma için kullanılan bir SIFT ve bir SURF algoritması öneriyor. Bu süreç, Kelime Çantası özellikleri, birleşik bir boyutsal histogram içindeki her bir eğitim görüntüsünün ana noktalarını ana hatlarıyla çizilen vektör niceleme tekniği kullanarak çalıştırılır. Bir eğitim ve test için radyal taban işlevi (RBF) üzerine kurulmuş çok katmanlı Destek Vektör Makinesi (SVM) sınıflandırıcı içine yerleştirilir. Öz çıkarma için bir görüntü işleme aracı ve aracı olarak açık CV C ++ kullanılır. Bu tezde, SVM tabanlı RBF çekirdeği üzerindeki SIFT performansı, SVM tabanlı RBF çekirdeği üzerindeki SURF ile karşılaştırılmıştır. SVM-RBF çekirdek sistemi ile SIFT kullanımının% 98.75'lik bir doğruluğa, ve SVM-RBF ile SURF' kullanımın % 97, 5'lik bir doğruluğa sahip olduğu bulunmuştur. Bir bilgisayar kullanıcısının ağda başka bir bilgisayar tarafından bir servis istediği ve sağladığı iletişimin istemci / sunucu modelini kullanan TCP / IP'yi kullandık. İstemci sunucu ağı iletişim sistemi, uygulama arabirimi için java programlama dili kullanılarak sağlanır ve ağ özelliğini algılamak ve sistem imgesini ve sistemin önemli verilerini depolamak için java seri hale getirilebilir arabirim yöntemini kullanır.

Anahtar Kelimeler: İmza tanıma, SIFT, SURF, SVM-RBF çekirdeği, BOW (sözcük paketi), Açık CV C ++, TCP / IP.

To Dear My Father, Mother, Sister, Brothers, and Friends

ACKNOWLEDGEMENTS

In the name of Allah, the Most Gracious and the Most Merciful, 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 am extremely grateful to my guide, Prof. Dr. NURAN DOGRU, for her attentive supervision and regular support. She spent her valuable time to assess the project work and give many positive comments. This project work would never have become a reality without her encouragement, advice and exceptional support.

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

At this Juncture, I thank of my parents whose selfless, sacrificial life and their great efforts with pain and tears and unceasing prayers has enabled me to reach the present position in life.

Last but not least, I want to express my deepest gratitude to my brothers, sister and friends for their love, understanding and their support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLE

Page

LIST OF ABBREVIATIONS

Open CV	Open Source Computer Vision
SIFT	Scale Invariant Feature Transform
SURF	Speeded-Up Robust Features
SVM	Support Vector Machine
RBF	Radial Basis Function
HMM	Hidden Markov Model
GWT	Gabor Wavelet Transform
MLP	Multi-Layer Perceptron
PCA	Principal Components Analysis
ED	Euclidian Distance
RBP	Resilient Back propagation
OCON	One-Class-One-Network
DWT	Discrete Wavelet Transform
DTCWT	Dual Tree Complex Wavelet Transform
RCWF	Rotated Complex Wavelet Filters
SDWT	Standard Discrete Wavelet Transform
COG	Centre of Gravity
RGB	Red, Green, Blue
BOW	BAG-OF-WORDS

FAR	False Acceptance Rate
FRR	False Rejection Rate
HSV	Handwriting Signature Verification
AER	Average Error Rate
IDE	Integrated Development Environment
API	Application Programming Interface
SYN	Synchronize Sequence Numbers
TCP/IP	Transmission Control Protocol/ Internet Protocol

CHAPTER 1

INTRODUCTION

Behavioral biometric stands for a signature that translates the ballistic movements of the signer for its selected signature. Compared to observable features like the finger-print, face, iris or a signature generally shows a higher degree of security and time variability. It is extracted from the Latin word "Signer" which stand for "Sign". For a long time, signatures have been used as an important element in authentication of any person's identity who is felicitating the document. Explaining of an individual in a transaction upon his authentication approach is a known proof by signature. A Signature recognition system has been stored inside two classes: a static approach (also referred to as Offline) which performs an extraction of the aspect which has a relation to the information and the dynamical approach (also known as Online) with time has a relation to information[1].

An offline signature recognition has images of signatures in bank checks and official papers utilized to verify and are helpful in automatic signature verification. The systems of offline signatures can be more reliable and simple to implement compared to the dynamic systems in various parts of the world; anyway, in a considerable degree, it is harder than dynamic verification because of the loss of dynamic information. In this study, the issue is dealing with an off-line or static signature recognition approach. There are a lot of challenge in the Offline signatures which is harder than On-line since the properties are taken from the static two-dimensional image of signature. It is known that a library of programming performance is identified as Open CV, in other words, Open Source Computer Vision (Open CV) which is generally directed at real-time computer vision the terms Scale Invariant Feature Transform (SIFT) and Speeded Up Robust Feature (SURF) algorithms, in computer vision, are a robust traits discovered and described approach for local merit

in pictures. SIFT takes distinctive invariant traits from pictures which are independent of image scaling, rotating and translation. SIFT has been applied in various applications such as an image mosaic, stitching, recognition, and retrieval [2].

In this study, an Offline signature identification with the use of Support Vector Machine (SVM) basic on Radial Basis Function (RBF) kernel with SIFT and SURF algorithm to offer a system for voting online. This system provides a solution in facilitating the remote election process.   We suggested SVM which is a new learning approach proposed by V. Vapnik and   C.Cortes[3].

In other words, SVM measures how complex the hypotheses are based the margin separating the data. This research is also used TCP/IP that utilizes the client/server structure of communication when a user asks and is granted a service from some other computer in the net-work. TCP/IP is initially a point-to-point, which refers to the fact that every communication is from a point in the network to some other point or host. Client/Server networks have some apparent benefits compared to the peer-to-peer type. It's simple finding files and resources due to the fact that they're kept in the server. In addition, they have much higher level of security. Each username and password is stored in the same data-base (i.e. on the server), and individual users are not capable of using the server in a form of a work-station. The server holds the data-base of user accounts, pass-words, and accessing privileges [4]

1.1. Signature Recognation

Signature is preferable among different biometrical methods due to the fact that it's the widely acceptable manner to identify a person in the everyday activities like the automated bank transaction, electronic fund transferring, document analyzing, and access control. Automated verification of signature is an area of research attempting at the creation of reliable online or offline systems that is capable of verifying human signature. Identification of signatures is a behavioral biometric Fig.1.1 shows handwriting signature. It might be operated in 2 different manners: static and dynamic.

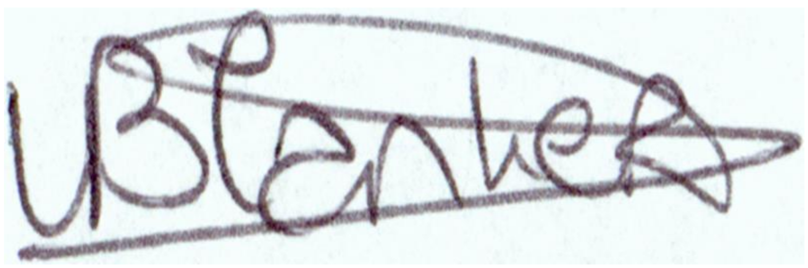


Figure 1. 1 Handwriting signature

1.1.1 Static

In the static mode, individuals write down their signatures on paper, digitize it via an optic scanner or a camera and the biometrical system performs the recognition of the signature analyzing its form. This mode is referred to as―Offline as well.

1.1.2 Dynamic

In the dynamic mode, individuals write down their signatures in a digitizing tablet that obtains the signatures directly in real time. One more possibility is the obtaining via of stylus-operated Portable Data Assistants. Dynamical identification is also referred to as ―Online.

1.2. Problem Statment

The aim of this work is to design and develop an Offline signature recognition using SVM basic on RBF kernel with SIFT and SURF algorithm to offer a system for voting Online by using signature. This system provides a solution in facilitating the remote election, process.

1.3. Objective

1. To study the Literature of signature biometrics and determine the hiatuses.
2. To study the present method to probe what their limitations are
3. To develop and improve two methods to compare together.
4. To offer a system for voting online.

1.4. Different Biometrics Technique

Biometrics is defined as measurable physiological and/or behavioral qualities that can be applied to verify the identity of a person.

They comprise the following:

1) Iris survey
2) Facial recognition
3) Fingerprint investigation
4) Hand geometry
5) Retinal survey
6) Signature investigation
7) Voice investigation

1.5. Scope

This thesis deals with static images of handwritten signatures. The genuine signature samples taken from data base casia. SIFT and SURF features are used as signature image descriptors. Scanned signature images should be tidy and clean. Images should be free of noise, because the noise makes difficulties to analyze the signature. Dissimilar signatures causes problems. So, signatures of a same person must maintain a similarity.

1.6. Backround Concept

There have been a lot of work done by the researchers in this field. Signature recognition is one of the most important authentications to prove the person’s identity: the bank-cheque purpose, the checking purpose and any lawful or formal purpose is based on the signature verification. The aim of this thesis is, time reduction and increase in precision. In the biometry signature, signatures proved by computers is an active research field. During the past few years there have been more researches concerning the verification of online and offline signatures. Online verification system gives more accurate result than that of offline signature but in some time or situation persons are not present physically. That’s why offline signature is more important than online signature. If we had a look in the past experimental work, the result is not so good and the success rate is not satisfactory. This research is concerned with developing of some unique features and methods that are applied for training and testing purposes, in order to obtain more accurate result. After training and testing we take images and save as it, and we use TCP/IP (client-server) protocol to send and receive data. In a 2-level client/server model, the client performs the communication in a direct way with the data-base server. The application or business logic either stays on the client or on the data-base server as stored procedures.

CHAPTER 2

LITERATURE REVIEW

Introduction

In this chapter the detailed survey of current literature on signature recognition and verification based on SVM, SIFT, SURF, Neural Networks, Discrete Wavelet Transform, Hidden Markov Model (HMM), Statistical and Distance Classifiers and Fuzzy Logic Based Approaches are given. The issue concerning the static signature recognition and verification has been the subject of interest in the past decade due to its promising applications in bank transactions and security systems.

2.1 Support Vector Machine (SVM)

It is a machine learning algorithm which uses a high dimensional space of features and find differences between classes of specific data for the generalization of the unseen data. The system in [5] utilized global, orientation and grid properties of the signature and SVM for classifying and verifying. The data-base of 1320 signatures is utilized from 70 writers. 40 writers were utilized for training with every one of them signing 8 signatures, i.e. a total of 320 signatures for training. For initial testing the method utilized eight genuine signatures and eight forged ones achieves False Rejection Rate FRR 2% and False Acceptance Rate FAR 11%.
S. Audet, P. Bansal, et al [6], designed offline signature verification and recognition with the use of SVM. They utilized global, directional and grid properties of signatures. Virtual SVM (VSVM) has been utilized for verifying and classifying the signatures and (FAR) of 16.0% and (FRR) of 13.0%.

Ozgunduz et al. [7]suggested an offline signature verification system with the use of SVMs. The authors utilized SVMs for detecting random and skilled forgeries. They utilized the extracted global geometrical properties, orientation properties and grid properties for support vector machine classifier. In the experimentations, a comparison between support vector machine and artificial neural networks was done.

Utilizing a support vector machine with RBF kernel, an FRR of 0.02% and an FAR of 0.11% were reached.

In [8] the authors, Vahid, Hamid and Reza introduced a new approach, Radon Transform for offline signature verification. In this suggested approach Radon Transform was used as a property extractor and they also used SVM as a classifier. And locally Radon Transfer also used for detection of line segmentation. In this method there were two major steps, one is pre-processing and another is feature extraction. They took datasets of 20 classes (Persian) and 22 classes (English), after
experimenting with these classes individually the authors compared the methods. In pre-processing stage first they scanned all the signature images. Then they binaries the scanned images and after that the outer rectangular portion was selected and the margin of the signature has been removed. In color inversion process the signature image was inverted and the Radon Transform counts the pixel value. The method worked locally so the segmentation which was applied on those signature images. After that the feature extraction which included 4 steps was applied; detect the line segment, validation of existence of the line segmentation, extract and summarize the feature vector and finally normalized the feature vector. At last they calculated the FAR and FRR values.

Sigari and Pourshahabi [9], suggested an approach for signature recognition according to Gabor Wavelet Transform (GWT) as a property extractor and SVM as classifier. In their research, following the process of size normalization and noise elimination, a virtual grid was put on signature image and Gabor coefficients were calculated on every one of the points of grid. Following that, all Gabor coefficients were fed into a layer of Support Vector Machine classifiers as a property vector. The number of SVM classifiers equals the number of the classes. Every SVM classifier decided whether or not the input image belonged to a corresponding class. In that research, 2 experimentations on 2 signature collections were performed. They reached an identification rate equal to 96% on Persian signature collection and over 93% on Turkish signatures.

2.2 Scale Invariant Feature Transform (SIFT)

Ruiz-del-Solar et al. [10]used SIFT to extract local interest points from both query and reference samples to build a writer-dependent classifier. After extracting interest points from both images, they generated a set of 12 features, using information such as the number of SIFT matches between the two images, the processing time (as a measure of complexity of the matching process), along with other information from the transformations. They then trained a Bayes classifier with these features. Using skilled forgeries for training, they obtained good results on the Digital Signal Processing Group (GPDS) dataset.

Tong Liu, et al. [11], their suggested system benefits from the SIFT properties possessing strong robustness to the changes in expression, accessory, pose and lighting. One Multi-Layer Perceptron (MLP) based network is adopted as classifier of SIFT key point properties. The suggested classifier performed a classification of every key point into face ID then an ID index histogram counting approach implemented as the identification of the approach for facial image recognition.

In [12]defined as robust image descriptors. A signature data-base gathered which consisted of known writers' signatures and their forgeries. The sufficiency of the verifier was tested and specificity and the sensitivity were assessed for every taken test. It was noted that some of the writers had noticeable differences between 3 of their signatures in a way that even a forgery could belong to the interclass distances that can be resulting in an incorrect negative notification. Classifier had to be of high specificity and sensitivity degrees. In order to be capable of having a sufficient classifier test was chosen which had high specificity and sensitivity degrees. With the use of a range equal to 0.05 on each of the minimal intra-class and maximal intra-class distances as a threshold in a way that the minimal and maximal interclass distances have to be in that range. The efficiency statistics that were resulted from this test illustrated that the properties of SIFT could be utilized via Euclidean distances for off-line hand-written signature verification.

Ref. [13] focused on the matching of the SIFT properties between two images, and calculating the distance for SIFT property vectors for the evaluation of the degree of similarity between the original image and the resized one.

Ref. [14] improved (and examined) the local image descriptor that was utilized by SIFT Similar to SIFT, the descriptors used for encoding the salient aspects of the image gradient in the property point's neighborhood; on the other hand, rather than utilizing the SIFT's smoothed weighted histograms, Principal Components Analysis (PCA) were applied to the normalized gradient patch. These experimentations showed that the principal components analysis based local descriptors were more distinctive and robust to image degradations, and more compact than the standardized SIFT representation.

2.3 Speeded-Up Robust Features (SURF)

In [15], S. Ahmad proposed a new method for signatures segmentation from machine printed text with the use of a part based approach. The authors used speeded-up robust feature (SURF) for recognizing the machine printed texts from actual signatures. It performed an extraction of the keypoint/keypoints of interest from an image. SURF features were more reliable and useful, that’s why the authors used this feature. For this experiment they used Tobacco-800 dataset which was publicly available. From the Tobacco-800 set 10 documents which had machine printed text and signatures, were used for training. In [14], the authors used 128 bit descriptor for finding the similarity between various keypoints. The Euclidean distance measurement was utilized as a distance measurement.

Malik et al. [16] used SURF features to classify among genuine signatures, forgeries and disguised signatures. They first used SURF to extract interest points in the signature images, and used these features to assess the local stability of the signatures (i.e. find parts of the genuine signatures that are more stable over time). During classification, only the stable interest points were used for matching the number of key-points in the query image, and the number of matched keypoints were utilized for classifying the signature as original or forged.

Ref. [17] proposed expressions invariant facial identification via the detection of the fiducially points and using the (SURF) paired with Gabor filter. The proposed approach was tested via test images that had various expressions and found to be a more efficient performer over the traditional SURF algorithm.

In [18], a robust facial identification method was suggested. SURF algorithm was utilized to extract the property vectors with scale invariance and pose invariance from the facial images. After that, the PCA was introduced to project the vectors to the new property space as PCA-SURF local descriptors. Lastly, the K-means approach was used for clustering property points, and the local similarity and global similarity were afterwards combined for classifying the facial images.

H.Bay [19] proposed an innovative scale- and rotation-independent interest point detector and descriptor, coined SURF. In [19] they used a Hess matrix-based measurement for the detector, and a distribution-based descriptor. SURF has two properties: SURF-64 and SURF-128. In [19], authors used SURF-128 feature. It give much better result than regular SURF but it worked slowly. They performed a comparison of the results with the use of two different matching approaches, 1) according to the similarity threshold and 2) on the nearest neighbor ratio. SURF performed well in both the cases. In [19] they took 216 images of 22 items, the test images (116) have taken in different situation. The images were matching by the comparison of the intersecting point of reference image and intersecting point of test image. For these author used Euclidian Distance (ED).

2.4 Neural Networks

Huang and Yan [20]used Neural Networks [NNs] to classify between genuine signatures and random and targeted forgeries. They trained multiple networks on features extracted at different resolutions (grid sizes), and another network to make a decision, based on the outputs.

Gulzar et. al. [21] proposed an offline handwritten signatures recognition system which is based on neural network. The hand-written signatures recognition system was trained with scanned low-resolution signature images. In [11] the authors applied a low pass filter for reducing the size of the image before interpolation using the ED interpolation method. They utilized the NNs for their adaptive nature of learning by example to solve problems. This property made computational models such as those very convenient for many applications.

O. C. Abikoye et al, [22] introduced an offline signature verification and identification system with the use of NNs. In their approach they used a different way apart from the conventional way of writing an algorithm to solve a computer problem. They developed a system that was able to recognize handwritten signatures and verify its authenticity with the use of the concept of neutrons in human brain that is familiar with medical practitioners. The system was developed, tested and found suitable for its purpose and results were presented in dynamic images. All these techniques were not invariant to scaling and rotation.

The suggested system in [23] utilized structure properties from the signature’s edge, updated orientation property and some other properties such as surface area, length skew and centroid property where a signature was split to 2 parts and for every part an index of the Centre of gravity was computed with respect to the horizontal axis. For classifying and verifying two methods were compared the Resilient Back Propagation (RBP) NN and (RBF) with the use a data-base of 2106 signatures which contain 936 original signatures and 1170 forged ones. Those 2 classifiers registered 91.21% and 88 % true verifications.

The method of [24] tried combining on-line and off-line Handwriting Signature Verification (HSV). For stationary images the scale, rotation and displacement invariance was represented according to a normalized Fourier descriptor which had been obtained via repeatedly retracing the contour and the results of the periodical function represented as a Fourier series. For a dynamical image a speed function was utilized as a descriptor. The on-line retracing was compared to the off-line image sample. For classification a Multilayer Perception (MLP) NN used with a single input layer, one hidden layer and one output layer. The results that were illustrated were for dynamical image.

H.B. Kekre, et al. [25] proposed Morphological Pixel Variance Analysis approach for this issue. Where the morphological dilation was applied to the templates of signature with various structuring objects. The system designed using EX-OR template matching based fuzzy classifier and system showed about 94.94% precision rate.

Vaibhav Shah, et al. [26] utilized curve fitting algorithm for identification. A collection of shape-based geometrical properties and the distance based parameters of signature via the analysis of the polynomial equation. Nib lack algorithm applied for removing noise from the image. System found robust for semi-skilled signatures, however, its efficiency degraded with skilled forgeries.

L.Basavaraj [27] introduced an NN-based system for detecting random forgeries. Every one of the signatures represented with the use of global properties, grid properties, and texture properties. For each of those property sets, a specified 2-stage perceptron one-class-one-network (OCON) classification model was used. In the initial phase, the classifier makes a combination of the decision results of the NNs and the ED computed by 3 property groups. The outputs of the initial phase classifier fed a 2nd-stage RBF-NNs structure that makes the ultimate decision.

Odeh et al., [28] utilized the GPDS300 signature data-base given by the "Universidad de las Palmas de Gran Canaria", Spain for those willing to study the area of identification. The parameters for the signature that was genuine were between 85% and 100% similarity rate. Those were based on the approach of the hand-written signature identification that says that signatures present a degree of variance on every attempt from the same individual. In addition, if the percentage of similarity lies in the range of 75% and 85% then the measurements can be somehow uncertain. And if the percentage of similarity was lower than 75% then it's considered to be quite doubtful. The experiment was performed on an average of 200 signatures and at the end the results was as follows, training precision degree: 64% and error rate 36% testing precision percentage 78.8% and error percentage 21%. Concluding the approaches for signature verification and identification in this study they utilized the MLP NN. Four properties considered that there was skewness, eccentricity direction and kurtosis. The algorithm utilized for the training of the NN was the back propagation. Finally, they enhanced the percentage of error from 44.9% to 21.1%.

2.5 Discrete Wavelet Transform (DWT)

In [29], the authors Prashanth C. R. and K. B. Raja proposed a new approach for Offline hand written Signature Verification, which was depending on Angular Features (OSVAF). In [6], the Discrete Wavelet Transform (DWT) is applied as a noise removal.
And DWT was also used to improve the spatial domain features of the hand written signatures. By pre-processing the scanned image of the signature was skeleton and the images of the signature obtained in accurate signature area.

In [30]the authors proposed Persian signature, instead of using text different shapes were used and therefore different procedure must have been used to authenticate such signatures. In [30], an offline method was suggested based on image registration, DWT and Image Fusion process to identify and verify Persian signatures. DWT was the first process which was used to access the signature details, then different instances of the same person signatures were fused together to obtain the signature details and to improve the fusion process registration method was used. The ED was used in the classification phase to compare the features.

In [31], the authors proposed a novel method, rotated complicated wavelet filter, which captures data in 12 varying orientations for off-line signature recognition. In [31], to derive feature extraction of the signature, the authors used the Dual Tree Complex Wavelet Transform (DTCWT) and Rotated Complex Wavelet Filters (RCWF). And they also utilized Canberra distance method in identification stage for comparing features. In [31], there was a comparison between the RCWF and DTCWT. The experimental results proved that the signature identification rate of RCWF method was superior over the DTCWT. The method was also compared with Standard Discrete Wavelet Transform (SDWT) which captures information in only three directions.

Deng et al [32] suggested a signature verification system based on the **Wavelet** transform: first the image was preprocessed to obtain a closed-contour of the signature. For every pixel in the contour, the coordinates (x and y), and the tangential angle were recorded. Each sequence was represented as a one-dimensional signal, and then a DWT was used for decomposing the signal.

2.6 Hidden Markov Model (HMM)

The method of Justino et. al [33], utilized the grapho-metric properties, which were static properties like pixel density and the pseudo-dynamical properties represented via axial slant. They utilized grid segmenting and split the signature image to 4 areas every one of which has a column that contained cells with horizontal projection and vertical projection. Every column was converted into a characteristic vector that was given a numerical value. An Hidden Markov Model **(**HMM) was utilized for the procedure of learning and verification.

In [34], a system was proposed which utilized general properties only. A discrete radon transformation that was a Sino graph was calculated for every signature binary image at range between 0 and 360 that was a function of the overall pixels in the image and the intensity per specific pixel computed with the use of a non-overlapping beams per angle for X number of angles. Because of this periodicity, it was shift invariant, rotation invariant and scale invariant. An HMM was utilized for modeling the signature of every writer. The approach accomplishes an Average Error Rate (AER) of 18.4% for a collection of 440 original signatures from 32 writers with 132 skilled forgeries.

Coetzer [35], utilized Discrete Radon Transform as global property extractor and a HMM in a new signature identification algorithm. In this approach, The Discrete Radon Transform was computed at angles ranging from 0◦ to 360◦ and every one of the observation sequences was afterwards modeled via a HMM of which the states were arranged in a ring. For modeling and verifying signatures of every one of the writers one HMM was taken under consideration. Their system was rotation invariant and robust according to reasonable noise degrees. Utilizing a data-set of 924 signatures from 22 writers, their system accomplished a percentage of rate equal to 18% when only high-quality forgeries were considered and an EER of 4.5% in the case of casual forgeries alone. Those signatures were initially captured off-line. Utilizing another data-set of 4800 signatures from 51 writers, their system achieved an EER equal to 12.2% when only skilled forgeries were taken under consideration. Those signatures were initially captured on-line and after that digitally converted to static signature images.

2.7 Statistical and Distance Classifiers

The individuality of writers' hand-writing was mapped with the one of the signature in Srihari et al [36]. The writer signed in a predetermined area of 2x2 inch and rotation was normalized according to the horizontal axis. The gradient, structural and concavity were utilized in the form of image descriptors. The gradient found the image's local properties and the concavity found the correlation between the structural and the local properties. The verification model depended on the Bayesian classifier in the fact that it utilized mean and variance measurements for classification. The system utilized 2 data-bases of signatures with a total of 106 writers and 3960 samples and obtained FRR of 21.90% and 30.93%.

The system used in [37] utilized global descriptors and local properties. The approach divided the signature to areas (envelopes) and obtain the Centre of Gravity (CoG) of sub-area and the distance made by the COG and the strokes white-spaces. The learning algorithm that was utilized was C4.5 and the classification approach depended on a decision tree. The method used 100 original signatures and 300 forged ones from 20 people consisting of 15 Chinese and 5 people that provide English signatures. For each case more than 90% of successful verification was recorded.

A unique approach was proposed in [38]where different properties were obtained, including global properties such as image gradient, statistical properties obtained from distributing pixels of a signature and geometrical and topographic descriptors such as local correspondence for tracing the signature. The classification was involved with obtaining differences between the signatures of the same writer and getting a distribution in distance space. For any signature under consideration, the approach estimated a distribution compared to the available known samples and a possibility of similarity was estimated with the use of a statistical Kolmogorov-Smirnov test. With the use of only four original samples for learning the approach accomplished 84% precision that might be enhanced for up to 89% when the original signature sample size was increased. This approach didn't utilize the collection of forged signatures in the training/learning.

The approach in [39] utilized the geometrical center for property extracting. The center was found via vertical and horizontal image splitting. The signatures that were utilized were captured at various times in order to show the intra-personal changes. The classification was performed with the use of a Euclidean classifier model that was a measurement of difference between any 2 image vectors. For the testing 21 original signatures and 30 forged ones were utilized. A collection of nine signatures was utilized to train the model, FAR estimated are 2.08%, 9.75% and 16.36% for random, simple and skilled forgeries. The FRR for genuine signatures equaled 14.58%.

In [40], a system that adopts an expert examiner approach was used which employed a smoothness criterion. The basis was formed in that skilled forgery signature greatly resembled genuine one at a general scale but they were less smooth. They derived a smoothness index as a ratio of non-smooth segments to total extracted segments and combined it with global features like baseline shift, aspect ratio. Using a data-base of 1320 original 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 [41] used similar approach as [40] but used crossing approach and fractal technique method to extract the smoothness property which they combined with global properties. A minimum distance classifier was used for verification. An AER of 17.3% was achieved for a database of 55 writers, with 24 skilled forgeries and 24 original signatures for every writer.

The system introduced in Chih-Chieh Huang et al [42] used displacement extraction approach, where the displacement function between any two pair of signatures was the summation of the squared ED between them and a penalty which ensured the smoothness of the displacement function. Based on this displacement a measure of dissimilarity was obtained between the genuine and forged signature. A data base of 20 writers was utilized with 10 training signatures, 10 signatures for original set and 10 for forged ones. An AER of 24.9% is achieved. The ED was achieved when the mean vector and the variance were used for estimation.

Use of a set of contour features that can describe the internal and external feature of the signature was proposed in [43]. The verification was based on Mahalanobis distance classifier. The training and testing were done through leave-one-out method. A data base of 20 writers was utilized with 10 training signatures, 10 signatures for original set and 10 for forged signatures. An AER of 11.4% was achieved [42]. Mahalanobis distance was achieved when a mean vector and the full covariance matrix of a given class was estimated and trained.

2.8 Fuzzy Logic Based Approaches

In [44], global properties of the signature such as the skeleton of the pen trace and the structure of upper and lower envelopes were utilized as shape descriptors. Those were extracted via the sampling of upper and external points from the signature’s binary image. Areas of high pressure where the writer made more pressure or emphasis to be produced to a linear function which would be utilized to maximize the relationship between the vertical projection and the horizontal projection of the skeleton. For every one of the shape descriptors that have been mentioned above a multi-layered perception was given and the network was trained using an updated back propagation algorithm and the result of every one of the individual networks is combined using a fuzzy integral voter. Using a collection of 1000 signatures the method accomplished 90% correct verifications.

The researchers in [45] proposed the system which obtained angle properties which were modelled into a fuzzy model according to Takagi-Sugeno model. This model was extended for including structural parameters accounting for changes in the styles of writers and variations in mood and the inputs were improved for deriving numerous rules. This method accomplished more than 70% true verification.

CHAPTER 3

METHODOLOGY

Introduction

Computer vision typically deals with object identification in a way which is independent of scale, pose, lighting and affine distortion. In this study we used SIFT and SURF algorithm with SVM based on RBF kernel. The SIFT algorithm takes an image and converts it to a set of local properties where every property vector is distinctive and independent of any scaling, rotation or translation. The algorithm of SURF is based on the same ideas and phases as SIFT; but the details in every one of those phases are different. In this project the SIFT and SURF features are considered. The implementation is done using OpenCV C++. The method is a two-phase procedure with signature training and testing, and TCP/IP (client-server) protocol is used to send and receive data. In a two-level client/server model, the client communicates directly using the data-base server.

The processed used in signature recognition system can be explained using the following flow chart .Two stages can clarify and display aspect of the system shown in Fig. 1,

A) Training stage.
B) Testing stage.

The first one is comprised of five major steps:
1) Signature picture from a database
2) Image preprocessing
3) Keypoint detector and descriptor by SIFT and SURF
4) BOW
5) SVM-RBF kernel.

The second one is comprised of 4 basic stages:

1) Signature that is to be tested from a data-base
2) Image preprocessing
3) Key point detector and descriptor by SIFT and SURF
4) SVM-RBF kernel

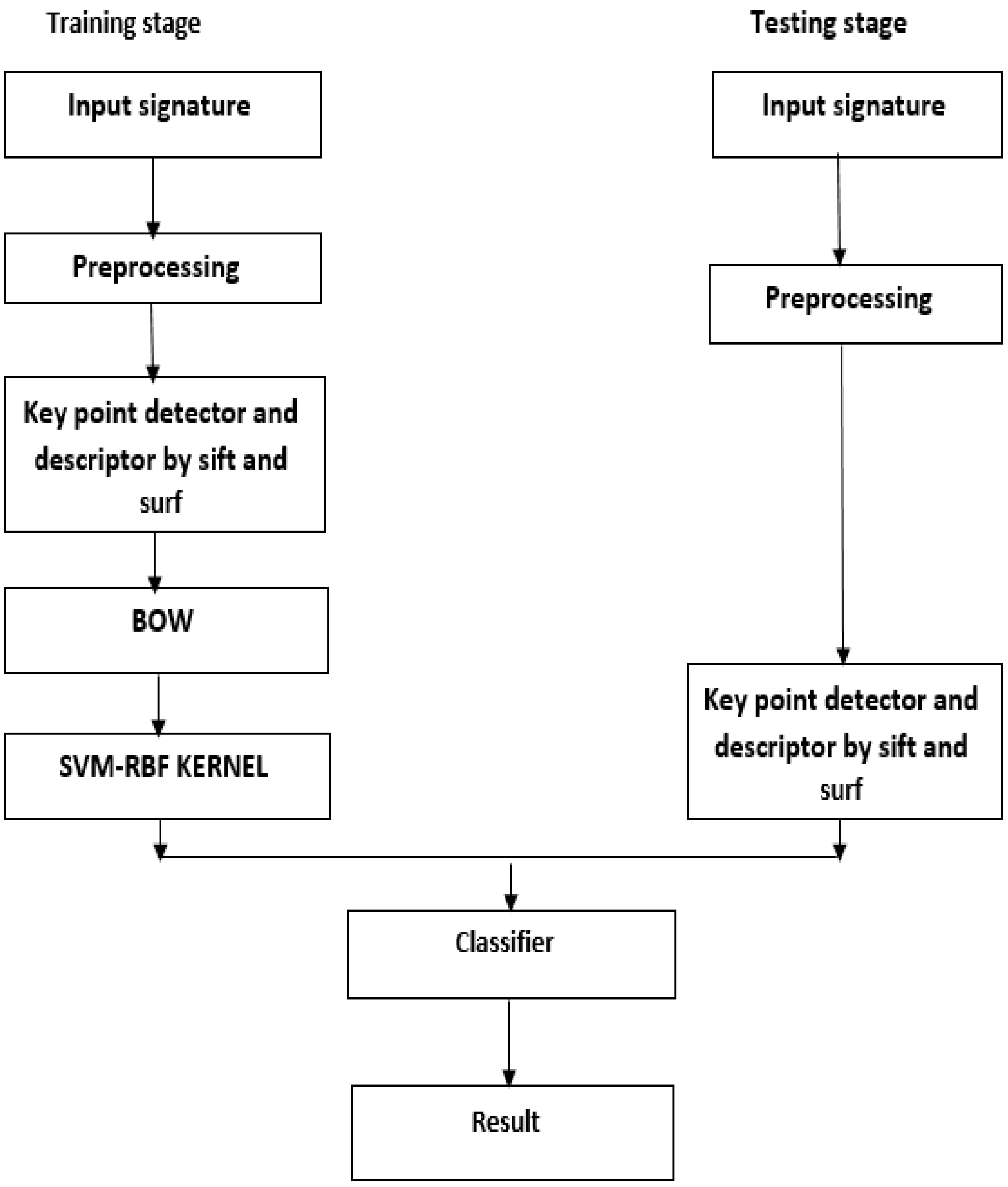


Figure 3.1: Flow Chart of signature recognition using SVM

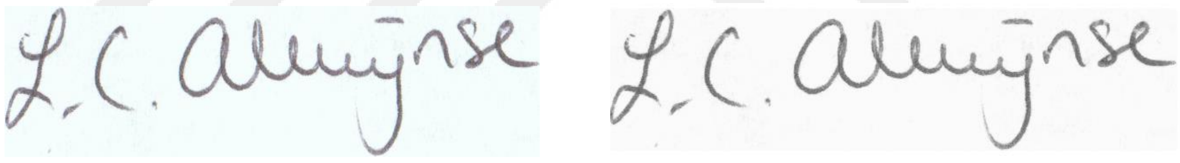
With RBF kernel.

3.1. Signature Database

The signature images are used for this study from the CASIA data-base. In order to train and test the signature identification system 240 signatures have been utilized. The signatures have been taken from 10 peoples. In order to train the system, we used 160 signatures per person 16 signature from within 10 people. In testing, we used 80 signatures for each person 8 signature from within 10 people.

3.2. Prepossessing

SIFT and SURF are originally very modern for common purpose identification and it doesn’t need an image pre-processing. A preprocessing step has been applied on signature images for obtaining better matching efficiency. The pre-processing step is applied image to gray scale. The aim is to enable signature ready for traits extraction both in training and testing phases .The prepossessing phase changes signature. When a color image is transformed into a black (0) and white (255) image, then this process is called RGB to grey conversion. Fig. 3.2 shows the image of an RGB to gray conversion.



(a) RGB image (b) gray image

Figure 3.2 Color image and black and white image

3.3. Extracting SURF Properties from Signatures

The SURF algorithm is generally made up of two parts: first, detecting the interest point then, performing description of interest points. A scale space representation is focused on 2 parts and both of them order differential operators. Being unique of the SURF approach, those procedures are quickly accelerated by using an integral picture and box has filters approaches. In the first part, locate the interest point in the image. The SURF features are calculated and points are matched between the input and out signatures. SURF detectors are seen in the significant points in the image, and descriptors are used to get the feature out of vectors at each interest point just as in SIFT. Hessian-matrix approximation is comprised by SURF to put and locate the key points rather than variant aspect of Gaussians (DOG) filter prepared in SIFT. This algorithm is similar to the SIFT. However, at the same time, it’s 3 times faster than SIFT in calculating the speed. About 64 dimensions in SURF will be utilized to save the time cost for the two traits which are the matching and the computation. Fig.3.3 shows (a) the original signature image and (b) the key points of the signature image.

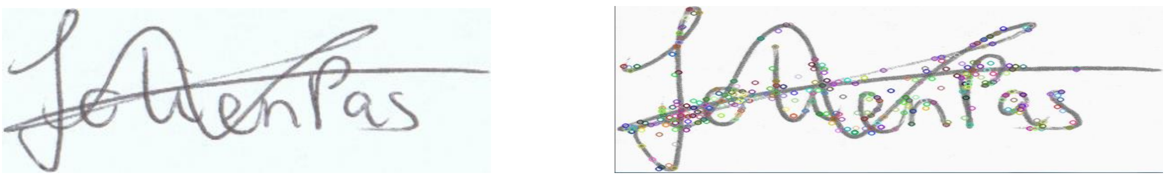

(a) Original image (b) Corrected image

Figure 3.3 Detected interest points of this image

3.3.1. SURF Detector

This algorithm utilizes the integral image for image representation in a way that the algorithm would be capable of proceeding with more speed. For a specific point, the integral image might be calculated via adding all the pixel values from the point to the coordinate. The integrated image may be represented using the following equation:

$$I(x, y) = \sum_{i=0}^{i\le x} \sum_{j=0}^{j\le y} I(x,y) \qquad (1)$$

A rectangle within an image can be expressed by using the integral image. The rectangle in Fig. 3.4 can be expressed by:

$$\sum A + B - (C + D) \qquad (2)$$

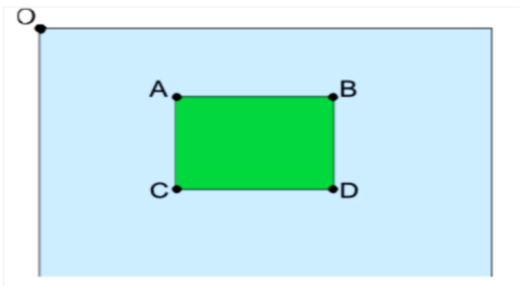


Figure 3.4 Region definition with the use of the integral images

The key points are located with the use of the determinant of Hess matrix. Gauss function is applied on an image and the Hessian for each of the sample points is calculated. The Hess matrix H(x, σ) can be expressed in the following form:

$$H(x, \sigma) = \begin{bmatrix} L_{xx(x,\sigma)} & L_{xy(x,\sigma)} \\ L_{xy(x,\sigma)} & L_{yy(x,\sigma)} \end{bmatrix} \quad (3)$$

Where L represents the Laplace of Gaussian and x represents the coordinates of the sample point and σ represents the standard deviation. And Lxx(x, σ) is the convolution of the Gaussian second order derivative $\frac{\partial 2}{\partial x2} g(\sigma)$ with the image I in point x, and similarly for Lxy(x, σ) and Lyy(x, σ).

Bay [19] proposed an approximation of the Hess determinant with the use of box filters that look similar to the Gauss kernels. The similarity is seen in Fig. 3.5. The black regions in the box filter representation are weighted as -2 for D_{xx} and D_{yy} , and the white regions are weighted as 1 for D_{XX} ,D_{xy} and D_{yy} the black areas for D_{xy} are weighted as -1.

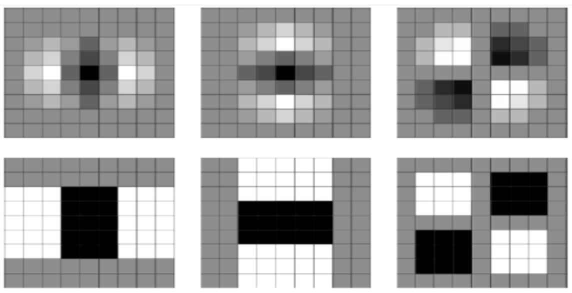


Figure 3.5 Gauss derivatives and box filters

The generation of the scale-space differs from that used in SIFT. Rather than sub-sampling the image, the standard deviation of the Gauss function and the size of the filter increases with each of the scales. The generation of the scale-space is illustrated in Fig. 3.6.

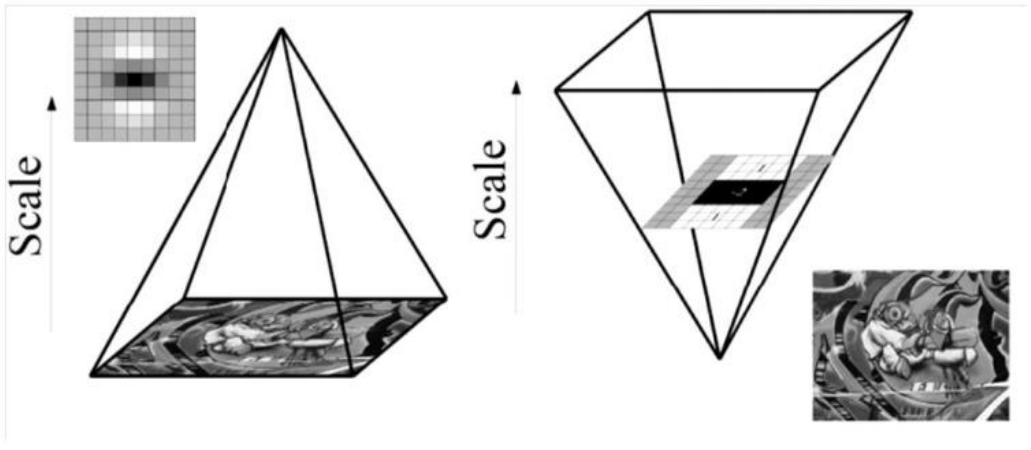


Figure 3.6 The generation of the scale-space

The size of filters increases by 6 within the 1st octave and the size is doubled at every one of the octaves, for example, the increment equals 12 at the 2nd octave and 24 at the 3rd octave. Fig. 3.7 illustrates the consecutive filter sizes for the 1st octave.

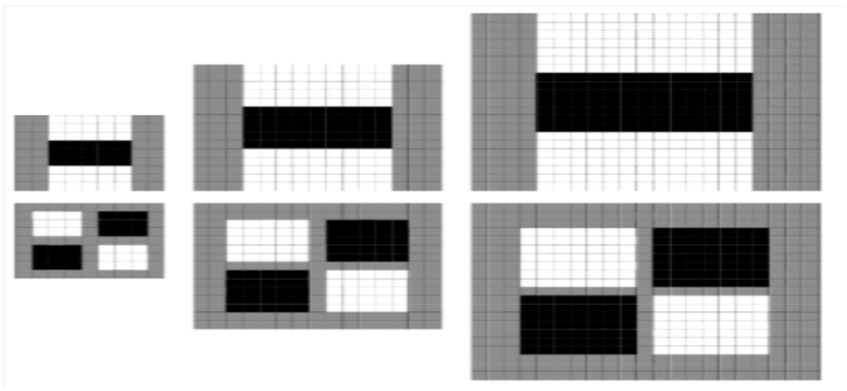


Figure 3.7 Filter structure.

3.3.2. SURF Descriptor

Initially, direction assignment is calculated for building a robust descriptor against various image rotations. The computations are calculated according to the scale of the interest point. The gradients of sample points are calculated via applying the Haar-wavelet on every one of the interest points and its adjacent points. The size of the neighborhoods is a circular region with a radius equal to 6s where s represents the current scale. The Haar-wavelet filters for responses in x and y directions are illustrated in Fig 3.8.

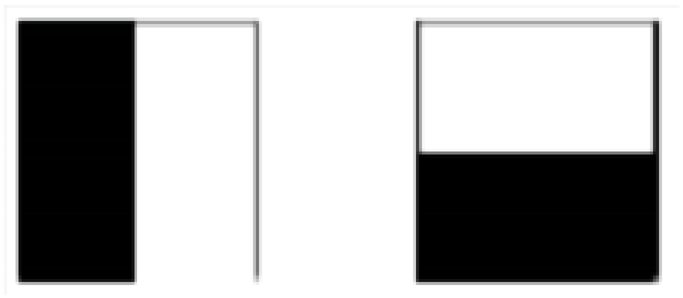


Figure 3.8 Haar-wavelet filters

Past computing the responses from the filters, they are weighted using a Gauss window. Standard deviation is chosen as 2.5s. The responses are summed inside an angle of 60◦ and the highest value denotes the direction. The scanning of responses is shown in Fig.3.9.

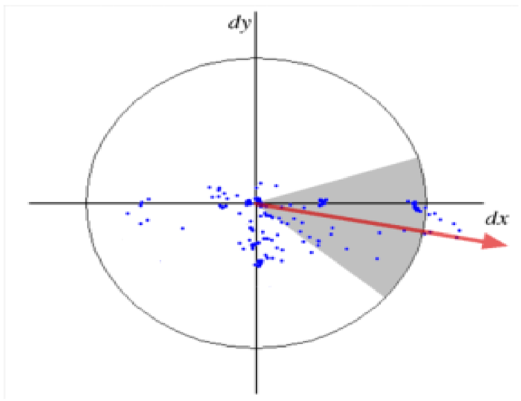
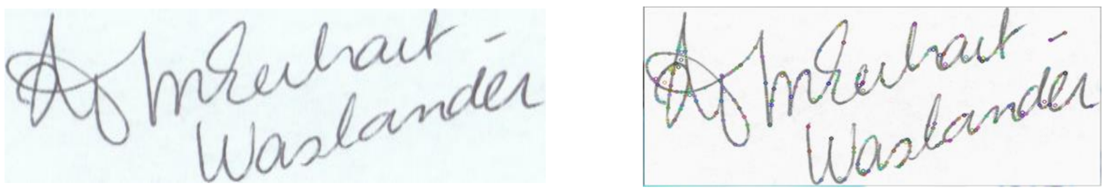


Figure 3.9 Orientation assignment

3.4. Extraction of SIFT Features from Signatures

We considered SIFT one of a number of computer vision algorithms which is clearly aimed at the extraction of distinctive and invariant properties from images .The novel image feature extraction was depicted at 1999 by someone is called David G. Lowe who study in British Colombia. We can called this method as SIFT [9]. Properties are obtained by making the SIFT algorithm independent of image scaling, rotating, and partially robust to modifying view-points and variations in lighting. We used 4 steps for extracting descriptors from an image and they are (1) the scale space extreme detecting, (2) accurate localization of keypoint, (3) orientation assignment and (4) key-point explanation. The 1st stage is to form the key points of images by using difference-of- Gaussian (DOG) function. We have two courses. The first one is candidate essential points which are prevented to sub-pixel essential points and left out to be found to be unreliable. The 3rd phase stands for the dominant orientations for every necessary point of the images .The last course is comprised by a descriptor for every essential point position focused on the picture gradients in their local neighborhood. We use the SIFT algorithm to get out the key points (vectors) for each try and test image. SIFT gets out repeatable property points from an image and produces descriptors standing for the texture surrounding the property points. SIFT approach moves an image data-set into a big set of local traits image vectors. Every one of the feature vectors has invariance are to scale, translation, orientation and noise. Fig.3.10 shows (a) the original signature image and (b) the key points of the signature image.



(a) Original image (b) Corrected image

Figure 3.10 Detected interest points of this image

3.4.1. SIFT Detector

Stable properties are extracted via searching the positions across numerous scales. Properties are detected with the use of Gauss filters with a varying variance according to a cascade method which makes the algorithm sufficient according to computations.

Step 1: Scale-Space Extrema Detection: The scale space of image, illustrated in Fig. 3.11, is generated via applying Gauss function on each scale with various values of standard deviation. The local extrema in scale space of Laplacian of Gaussian is treated as a key-point L(x, y, σ) is generated via convolving the Gauss function G(x, y, σ), with a source image I(x, y):

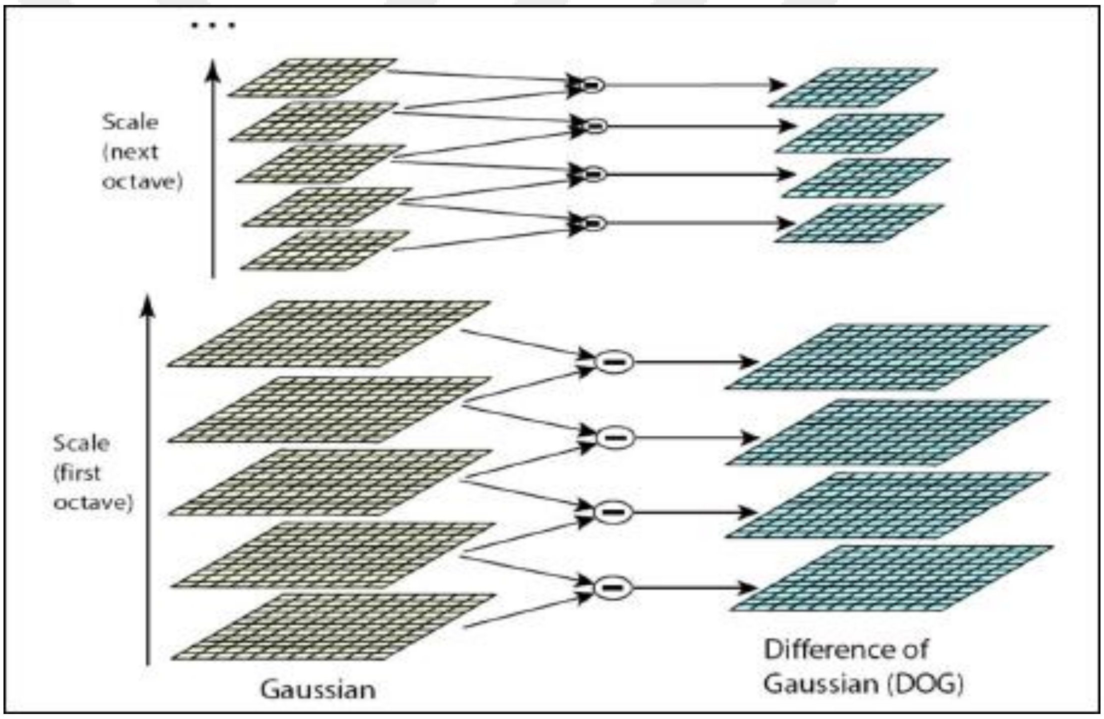


Figure 3.11 The construction of scale space

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \qquad (4)$$

Where * is the convolution in the *x* and *y* directions, the difference between two nearby scales, D(x, y), separated by a constant multiplicative factor k is given by:

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \qquad (5)$$
$$= L(x, y, k\,\sigma) - L(x, y, \sigma)$$

The key-points are identified as local maxima and local minima of the difference-of-Gauss signature images across the scale. Every pixel in the difference-of-Gaussian is compared with other eight adjacent pixels at the same scale and nine corresponding neighbors at the neighboring scales. In the case where the key-point is the local maxima or the local minima, it’s chosen as a candidate key-point, Fig. 3.12 shows the detection of the maxima and minima of the DoG in the scale space.

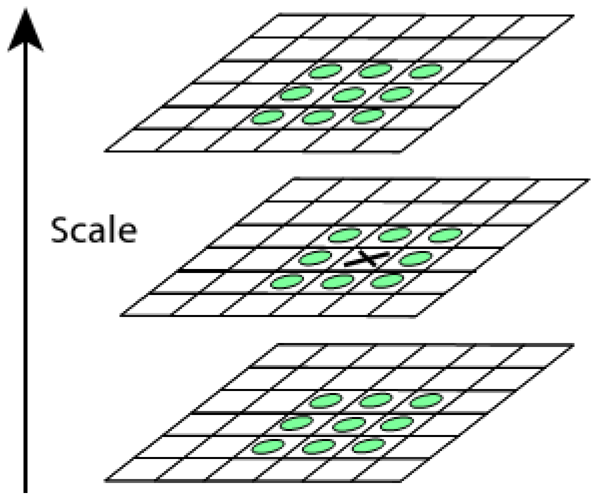


Figure 3.12 The detection of the scale space extrema

Step 2: Accurate Keypoint Localization.

For every one of the candidate keypoints identified, the interpolation of neighboring data is utilized to precisely locate its point. Key-points that have low contrast (i.e. the ones that are sensitive to noise) are dropped together with the responses poorly located along the edges.

Step 3: Orientation Assignment.

Every key-point is given one or more directions according to a local image gradients orientations. For determining the direction of the key-point, a gradient direction histogram is calculated in the neighborhood of the key-point with the use of the Gaussian image at the nearest scale to the key-points.

The impact of every one of the adjacent pixels is weighted by the gradient magnitude and a Gauss window with predetermined to be 1.5 times the keypoint scale. This contributes to stability [2]. Peaks at the histogram correspond to the dominant direction. Any key-point which is within 80% of the highest peak is utilized for creating a separate key-point. The direction assignment of every keypoint is found via calculating the gradient magnitude M(x, y) and direction $\boldsymbol{\theta}$ (x, y) of the scale space for the scale of that key-point:

$$M(x, y) = \sqrt{(K(x+1,y) - K(x-1,y))2 - (K(x,y+1) - K(x,y-1))2} \quad (6)$$

And

$$\boldsymbol{\theta}(x, y) = \arctan\frac{K(x,y+1)-K(x,y-1)}{K(x+1,y)-K(x-1,y)} \quad (7)$$

All the features of the key-point are measured with respect to the direction of the key-point. This caters for rotation invariance.

3.4.2. SIFT Descriptor

After finding the key-point's scale, location and orientation, key-point descriptors are calculated for achieving invariance to variations in illumination and changes in three-dimensional view-point. A Gauss window is estimated over the region of the descriptor and its neighbors. The points along the edge and the ones near it are weighted less so the gradients that are near the center give more valuable information. Thus, the impact of sudden variations along the edge is removed. The gradient magnitude of every one of the pixels is weighted via the Gauss window with σ that is 1/2 the diameter of the window.

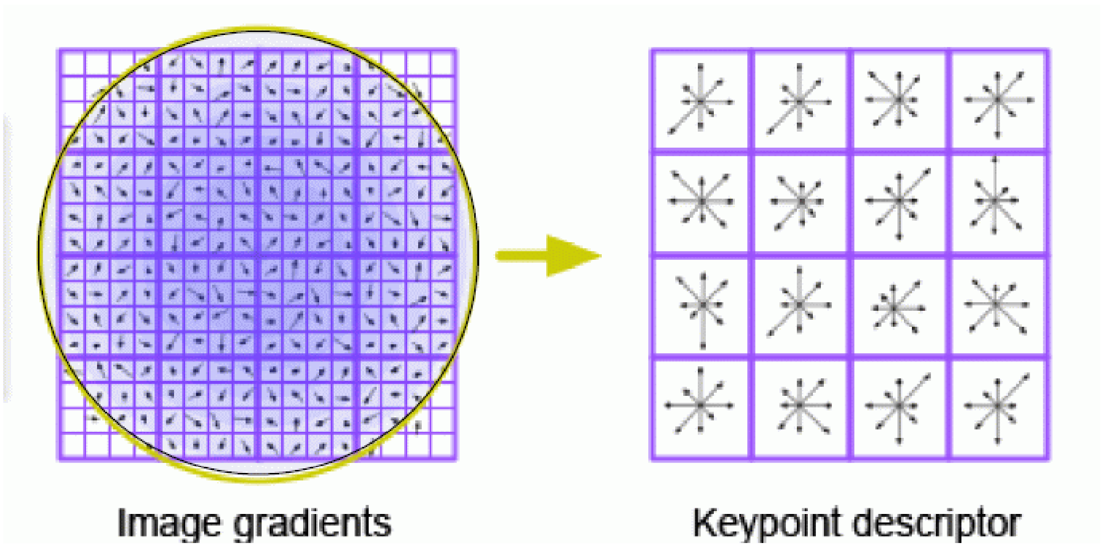


Figure 3.13 SIFT descriptor representation

16 orientation histograms are calculated within a 16×16 area and every one of the histograms is made with eight directions. Thus, the descriptor is made up of 4×4×8=128 elements. Fig. 3.13 illustrates the SIFT descriptor representation but the descriptor is calculated over the points within 8×8 area.

In the last stage, the descriptor is altered for decreasing the impact of variations in lighting. Each pixel value is multiplied with a constant when contrast variations in an image. The gradients are multiplied by the same constant as well. Via the conversion of the descriptor into a unit vector, any variation in image contrast will not be effective. Any variation in brightness will produce adding a constant to every one of the pixels. Nevertheless, the constant won't change the gradient values due to the fact that it will be discarded when pixel differences are computed to produce the gradient values. The gradient magnitudes are sensitive to nonlinear lighting changes so that the threshold (0.2) is set on the large gradient magnitudes in the unit vector and the vector is re-normalized to unit length.

3.5. Bag-of-Words Model (BOW).

We used here the method Bag-of-words model signature image classification. It is also usually referred to as Bag-of-feature (BOF) [46]. In this method a dictionary of smaller image segments of the image are formed and is done in such a way that this dictionary is referred back to when the test image needs comparison to be made.

BOF modeling is made by forming vocabulary of visual words from a group of region descriptors. It directs throughout some stages: area detection, region descriptor extraction, and quantization of vectors. In area detection, there are two kinds of area detection approaches key point - based area detection and dense sampled area detection. The first one produces patches randomly or uniformly, whereas the second one performs detection of the patches around key points with the use of invariant local property detector. The place of the regions of Key point-based are at salient points like corners and blobs. Therefore, local areas of interest are chosen and they're not found in homogeneous image. However, dense sampled areas are randomly chosen independently of image distribution or properties. The following stage is extracting the area descriptor. Those descriptors are obtained from a collection of detected areas. The area descriptors has to be described with invariant properties independent of the lighting, rotation, or other elements. A vector quantization is made for generating a visual vocabulary via making a collection of area descriptors taken from different images.

A property vector of every image is denoted by the occurrence of the visual words, Fig 3.14 illustrates the Bag of word vectors.

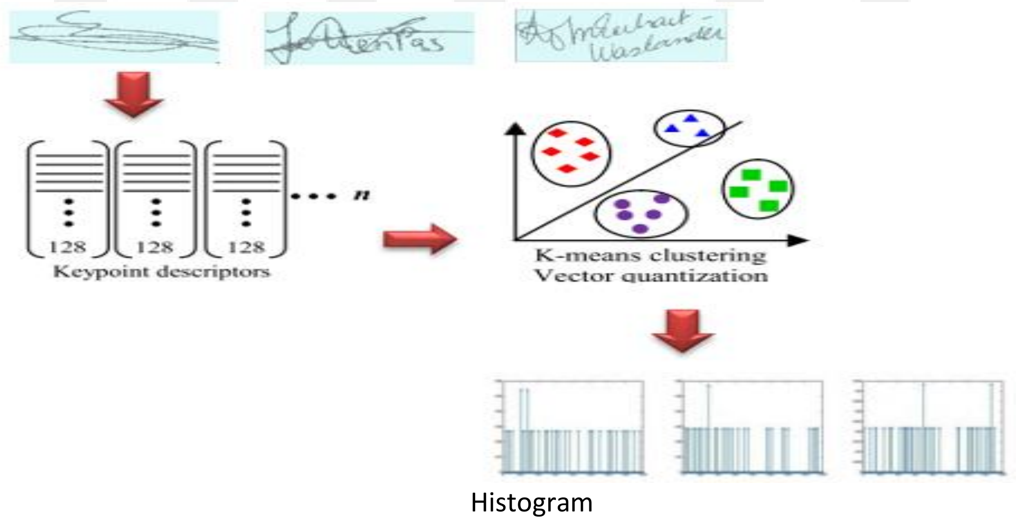


Figure 3.14 Bag of word Vector

There might be questions as to why we need to convert SIFT and SURF to BOW model to be analyzed. The first and the foremost reason is that the data distribution is so complex, complexity with respect to within class variance and inter class variance, that there are high chances that the classifier algorithms may not converge even with the help of kernel and this problem was faced SVM classifier and it applies to other classifier as well. If the within class variance is less and inter class variance is high then the distribution is less complex.

The BOW approach implementation stages:

Step 1: initially split the image into patches with the use of a uniform grid. In this situation, every one of the images is divided into uniform grid split into patches, in a way that every image may be represented with several patches. Then obtain the single scale SIFT and SURF features of the key-point in every patch, as illustrated in Fig (3.15) and (3.16).

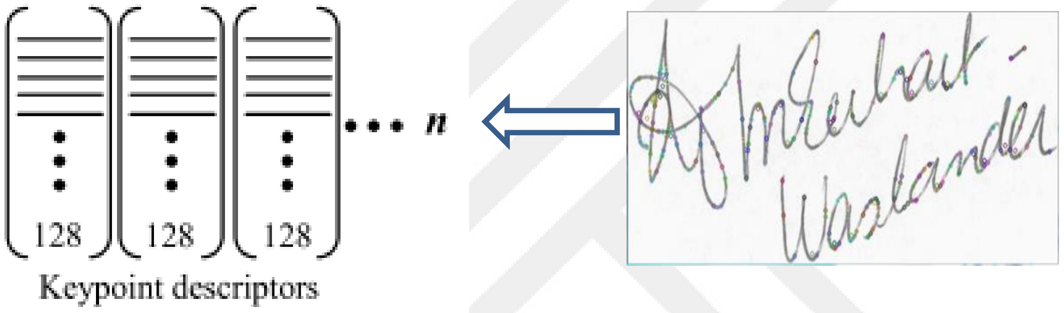


Figure 3.15 SIFT property extraction and description

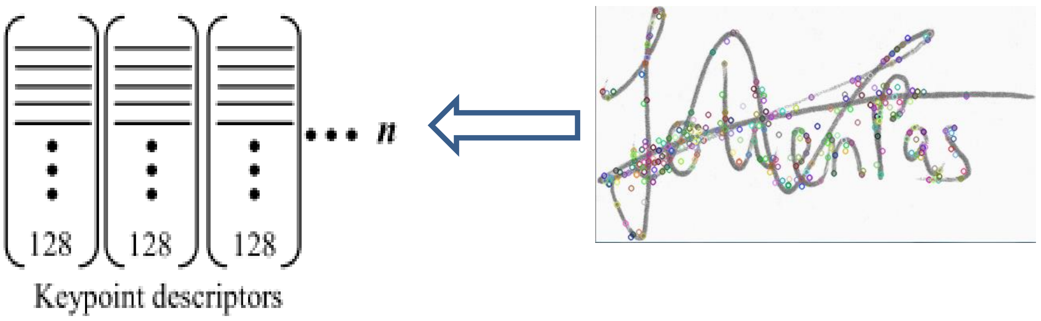


Figure 3.16 SURF Feature extraction and description

Step 2: Considering the fact that the size of the visual dictionary is 200 that is, there are 200 words in the visual vocabulary. Cluster the properties of each patch using the K-means clustering algorithm where K is equal to 200. Therefore, every clustering center can be extracted. The 200 centers (20-dimensional) are accurately regarded as the 200 words of the visual dictionary. Until now, constructing the dictionary is accomplished efficiently, as depicted in Fig (3.17).

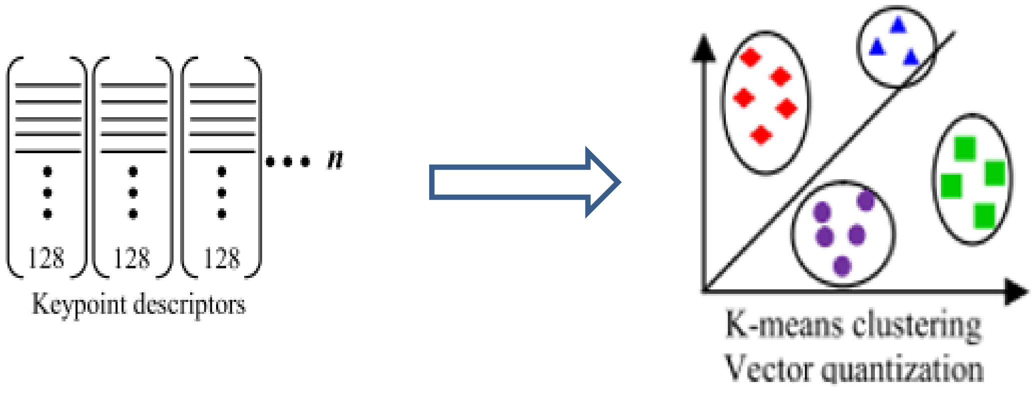


Figure 3.17 The construction of the dictionary

Step 3: Initializing a histogram h with 200 bins having the initial value of 0. Therefore, every one of the clustering centers corresponds to every bin in the histogram. For every one of the patches of an image, compute the ED between the patch and every one of the clustering centers. By comparison, it is concluded which clustering center is the closest to the patch. Make the bin corresponding to the clustering center add 1. Post computing every patch of the image based on the steps above, a histogram with 200 bins is obtained. After that it is normalized and using 200 d vector as the representation of the image. That is, the image is represented with the histogram that reflects the rate of occurrence of the visual words', as illustrated in Fig (3.18). Following the stages above are accomplished for all images, a support vector machine classifier is used for training and testing.

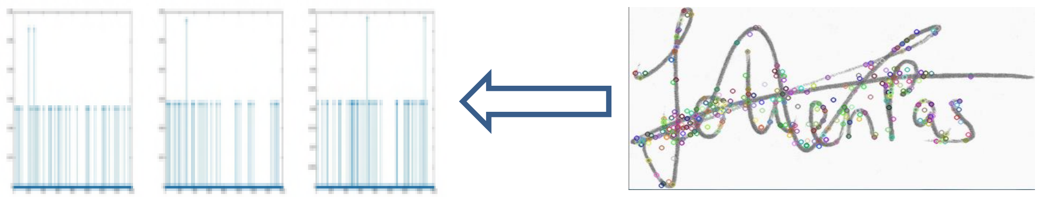


Figure 3.18 The representation of the image.

3.5.1. K-Means Clustering

K-means is an approach of vector quantization, and one of the simplest un-supervised learning algorithms which is capable of solving the common clustering problems. The process follows a simple and easy way of classifying a specific dataset via a specific number of clusters (for instance, k number of clusters) fixed apriority. The basic concept is defining k centroids, one for every one of the clusters. Those centroids have to be placed in a cunning way due to the fact that varying positions because varying results. Therefore, the optimal solution is placing them as far as possible from one another. The following stage is taking every one of the points that belong to a specific dataset and associate it with the closest centroid. When there aren't any points pending, the initial phase is done and an early set age is accomplished. Here it is required re-calculating k new centroids as bary-centers of the clusters that have resulted from the previous stage. After having those centroids, a new binding must be performed between the same points of the dataset and the closest new centroid. A loop has been produced. Resulting from this loop it is noticed that the k centroids update their location gradually until no variations happen anymore, i.e. centroids don't change their location finally, this algorithm has the aim of minimizing an objective function. The improving criterion in the clustering procedure is the sum-of-squared-error E between the items in the clusters and their respective cluster centroids $cen_{1,}$ ......cen_k , cf. formula.

$$E = \sum_{i=1}^{k} \sum_{o \in ci} d(0, ceni) \qquad (8)$$

The k-Means algorithm has a sensitivity to selecting the initial partition, so the initialization has to be varied. K-Means imposes a Gaussian parametrical design on the clustering result and in general works well on datasets that have an isotropic shape of cluster, due to the fact that it tends to generate compact clusters.

3.6. Support Vector Machine

It is the most efficient categorizing tool for supervised learning. The aim is separating the training data by a margin according to class labels assigned for every image. There could be more than a single possible line which is capable of separating the points, on the other hand, the line that has the biggest margin gives more precise classification outputs. The issue that support vector machine tries solving is finding an optimal hyper plane which properly classifies data points via separating the points of two classes as much as possible. Fig 3.19 shows margin selection:

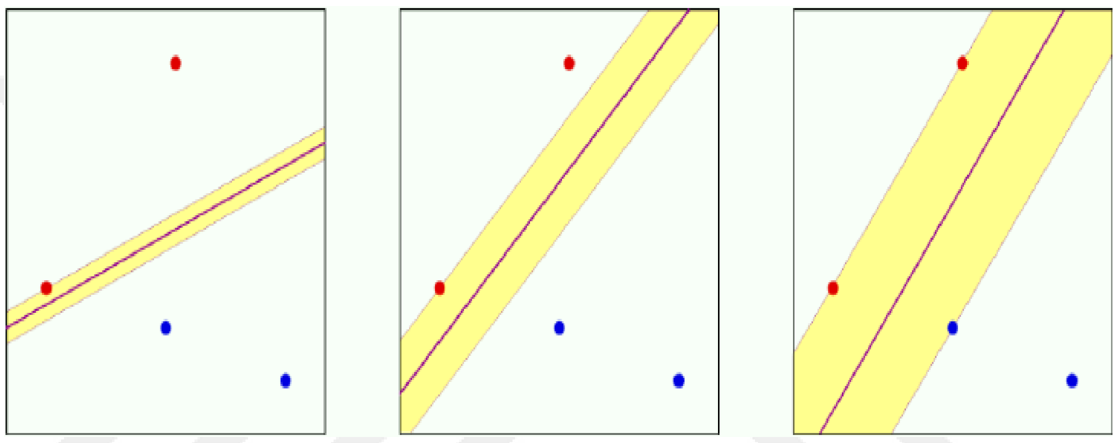


Figure 3.19 Margin selection

The margin can be defined as the distance from the plane to a specific point. Considering training data (x_i, y_i) for i=1 ….N, with $x_i \in \Re^n$ and given class labels $y_i \in \{-1, 1\}$, assuming that the set of labeled inputs, where -1 means that the input doesn't belong to that class and 1 means that the input belongs to that class. On the other hand, many images are utilized for training a classifier and the property space is highly dimensional. Thus, the data isn't linearly separable in the majority of the situations. The solution is applying a kernel function to every one of the points. SVM classifier has been used with RBF kernel because features used have big dimensionality depending on the number of vocabulary, and to achieve best result. The RBF kernel is given by:

$$K_{RBF}\,(x_i\,,y_j\,) = \exp\left(-\frac{\|y_j - x_i\|2}{2\gamma^2}\right) \tag{9}$$

Where γ is a tunable parameter. An example of the Radial Basis Functions classifier is illustrated in Fig. (3.20). the two sets of data are separated with the classifier and the support vectors presenting the margin, are circled.

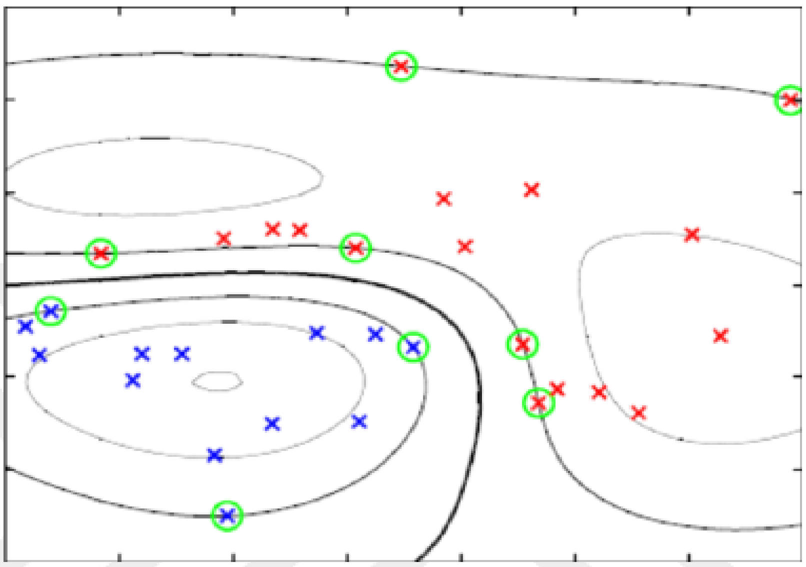


Figure 3.20 SVM with Radial Basis Functions kernel

The training images are labeled based on the annotations. SVM learns the way of separating the training data with the mapping labels. Each of C and γ are calculated with the use of train_auto () function in OpenCV. A main benefit of SVM classification is that it performs well on data-sets having a big number of attributes, even when there are only a few situations available for the training procedure. Nevertheless, many drawbacks of support vector machine classification include limitations in speed and size during each of training and testing phases of the algorithm and selecting the kernel function parameters.

3.7 System Architecture for Voting Online

The architecture for the suggested system is shown in Fig. (3.21).The system runs in parts on the computer (client) and on the server. The major components for the system are as follows:

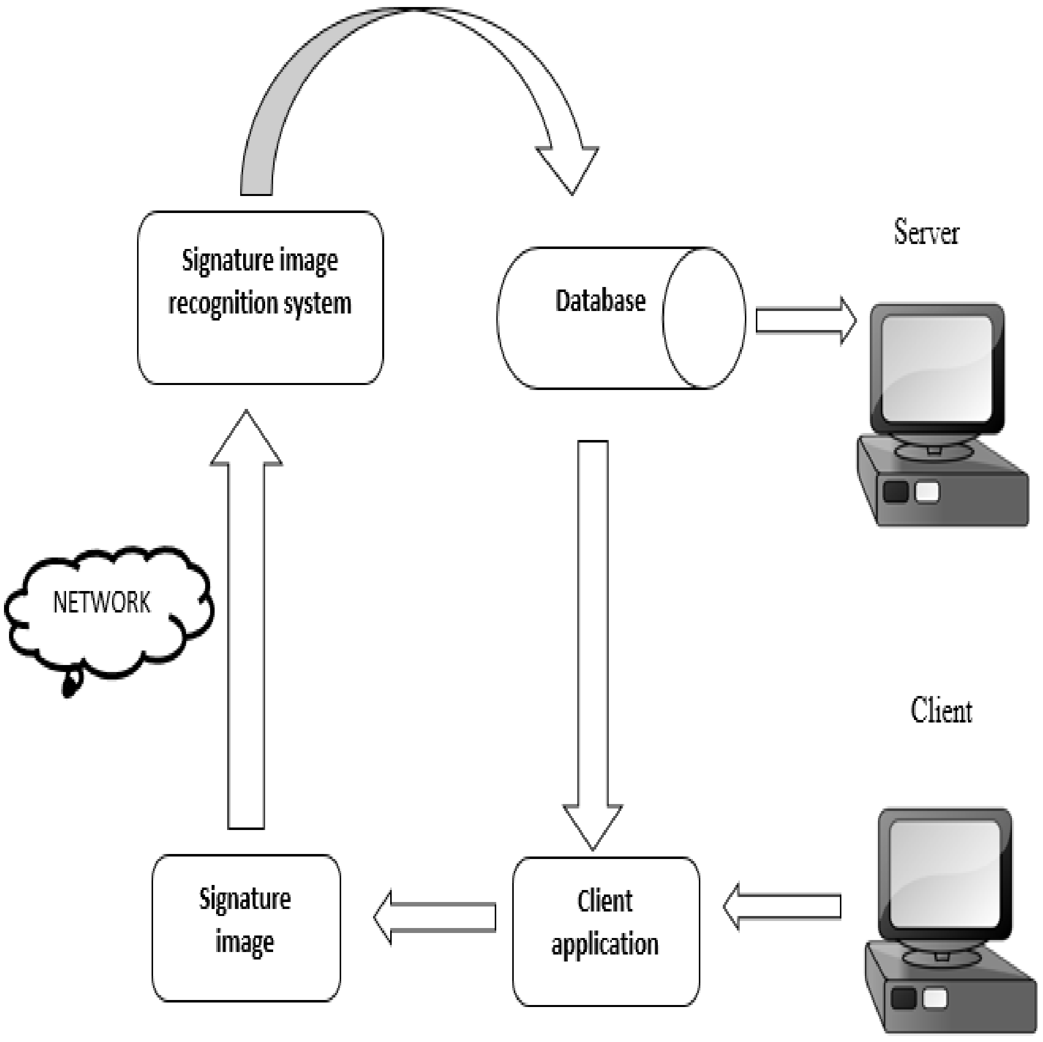


Figure 3.21 System Architecture

Computer Client component:

• NetBeans IDE

• Client interface application.

• Signature images

• Client communication protocol.

Computer Server Component

• NetBeans IDE

• Oracle database.

• Signature image recognition system

• Server communication protocol.

3.8. Computer Client Component

The application component on the computer client side is performed on a NetBeans IDE java platform. The computer client holds the responsibility of sending the signature image to the server. The first portion of the function is involved with signature image capture and matching between signature images and signature image recognition system for showing results, the client interface application obtained from the server for displaying the outputs.

3.8.1. NetBeans IDE

It is a software development platform that is programmed in Java. It gives the applications the ability of being developed from a collection of modular software elements known as modules. Applications that are based on this Platform, including the NetBeans Integrated Development Environment (IDE), could be extended by third party developers. NetBeans IDE is an open-source IDE. It supports developing all Java application kinds (Java ME, Java SE (including JavaFX), web, EJB and mobile applications). In addition to other properties are an Ant-based project system, Maven support, refactorings, and version control (supporting CVS, Sub-version, Git, Mercurial and Clearcase). Fig (3.22) provides a glimpse into the NetBeans IDE application framework.

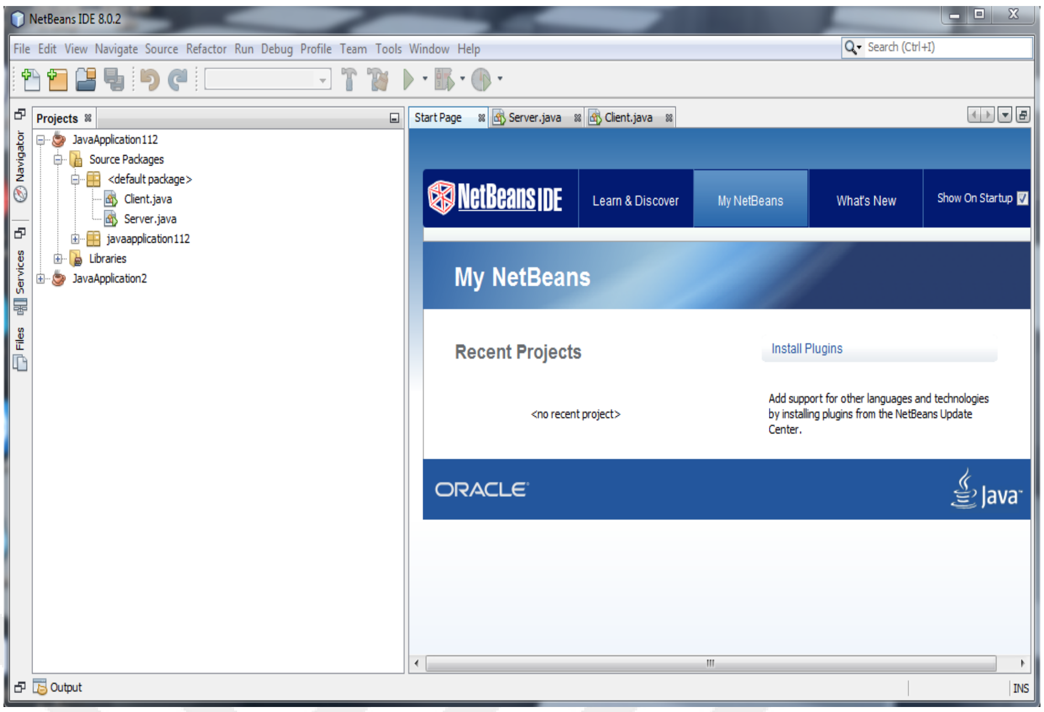


Figure 3.22 NetBeans IDE java Application Framework

3.8.2 Client Interface Application

Client Interface application is a NetBeans IDE java application that executes in the NetBeans IDE runtime, as illustrated in Fig (3.23).

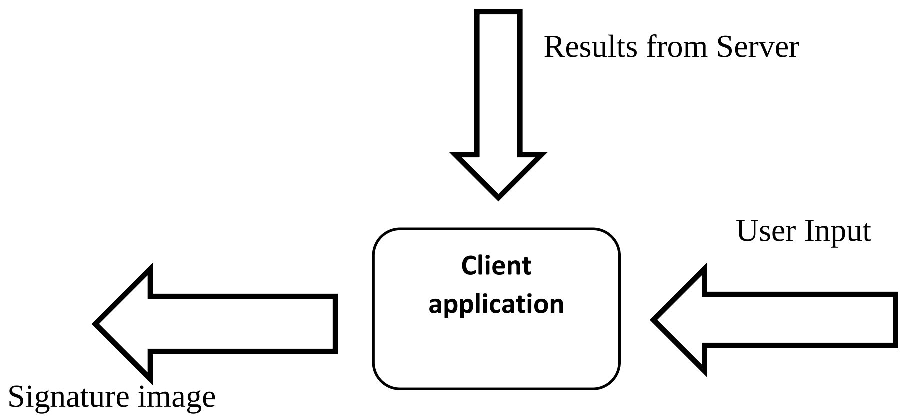


Figure 3.23 Client Interface Application

3.8.3 Client Communication Protocol

The communication Protocol is part of the signature image recognition application. The function of the protocol is to provide a standard mechanism for data exchange between the server and the client application. Here we use the standard TCP/IP protocol for communicating with them. For our application, NetBeans IDE provides standard APIs to be invoked for implementing this protocol.

3.9 Server Component

The server side of the application holds the responsibility of implementing the sending data. Prior to a sending data, it is necessary to produce a reference data-base. This data-base stores the Metadata for the signature image. First, we gather the signature images to populate our reference database. In Fig (3.24), we present the steps for populating a database for signature images.

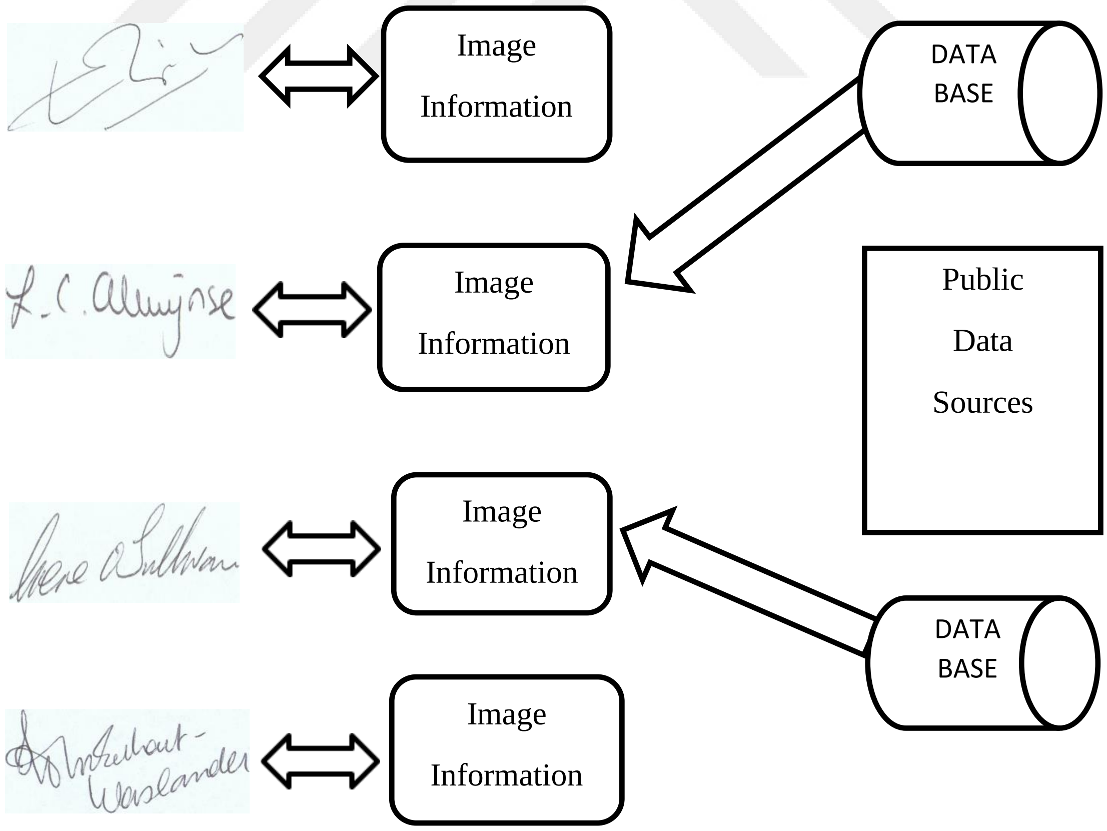


Figure 3.24 Step One for Populating Database - Signature Images

In the second step stored in the oracle database, Shown in Fig (3.25).

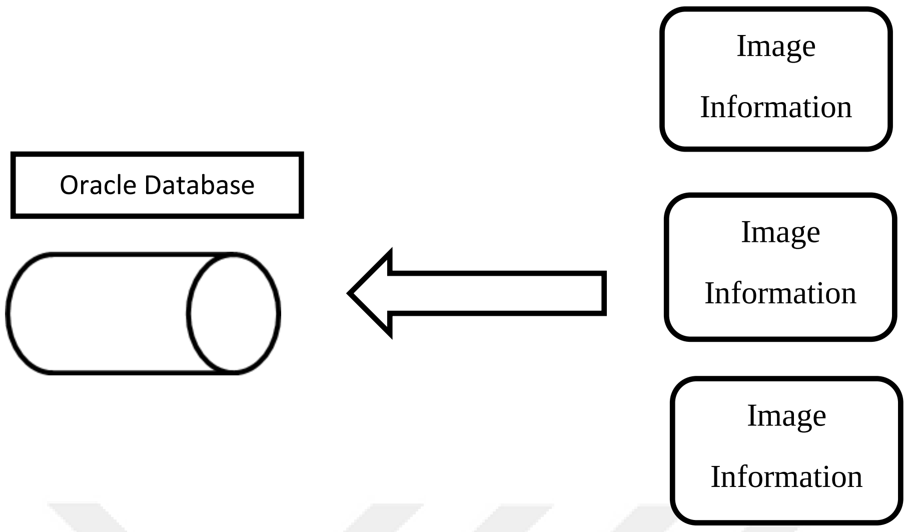


Figure 3.25 Step Two stored information signature image in the oracle data-base.

Here the signature image that has been obtained from the client side undergoes matching with the stored signature image set. The server side implements a database for storing the Meta data associated with the image collection. After the best match is selected, the server application retrieves the Meta data of the matched image, which includes relevant information about the image. then meta data can include the ID_Num,pName,fath_name,mother_name,Age,address,Blood_type,occupation,birth _ date . The server application then encodes this information and sends it back to the client application. We now will discuss the server side components.

3.9.1. NetBeans IDE

It was discussed in section 3.8.1.

3.9.2. Oracle Database

The database on the server side carries the relevant information about the signature images. The schema for the database will depend on the target application type. An Oracle data-base is a set of data treated as a unit. The aim of a data-base is storing and retrieving related data. A data-base server is the key to solve the issues of data management. Generally, a server reliably performs the management of a large amount of data in a multi-user environment so that a big number of users could.

Access the same data at the same time. All those functions are performed while providing high performance. In addition, a data-base server prevents unauthorized accessing and gives optimal solutions for failure recovery. In Fig (3.26) we show a sample database schema.

Image Information
ID
NAME
FATHER NAME
MOTHER NAME
BIRTH DATE
AGE
JOB
BLOOD TYPE
ADDRESS

Figure 3.26 Sample Database for Server.

3.9.3. Signature Image Recognition System

As discussed in section 3.1 to 3.6 in detail, the first step in the server side processing is to find the most appropriate match for the signature images. The application server forwards the signature images to the application.

3.9.4. Server Communication Protocol

The server communication with the client also occurs over TCP/IP. The server communication components receive the information .This then is sent serially to the client for additional processing. The Java NetBeans IDE architecture provides standard APIs to make the communication occur seamlessly.

3.10. TCP/IP (Transmission Control Protocol/ Internet Protocol)

It is a two-layer program. The higher layer, the TCP, holds the responsibility for managing the assembly of a message to smaller packets sent via the Internet and received by a TCP layer which re-assembles the packets into the original message. The second layer, the Internet Protocol, holds the responsibility of handling the addressing portion of every packet so that it reaches the correct destination. Every gate-way computer on the network performs a checking of this address in order to determine where to resend the message. Even though some of the packets from the same message are forwarded to different directions than others, TCP/IP utilizes the client/server architecture of communication where a computer user asks for and is granted a service from some other device in the net-work. TCP/IP communication is basically a point-to-point, which means that every communication is from a point in the network to one other point or host computer. The TCP/IP protocol stack models a set of protocol layers for net-works and systems which is beneficial for allowing communications between any types of devices utilized in communication. This layer model includes 4 separate but correlated layers, as illustrated in Fig. (3.27). Those four layers are necessary as the Internet protocol suite is based on them. The layers of net-work and transport, and the application layer are the most significant layers of TCP/IP architecture [47]. Those layers define the way of interfacing the net-work layer with the data link and physical layers, but it is also true that this is not in a direct way concerned with those two layers.

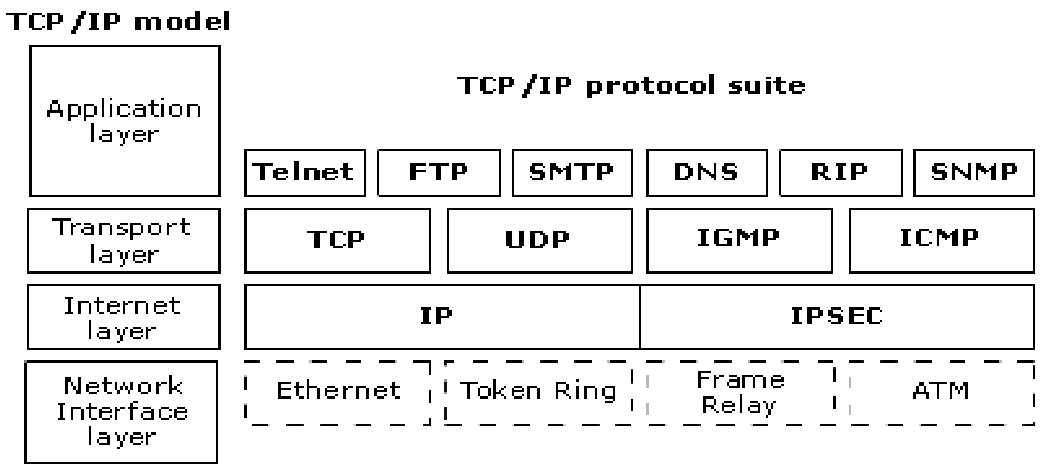


Figure 3.27 TCP-IP Protocol Suite.

- **The Application Layer:** it is the domain within which applications produce data and communicate this data to other applications on the same host or another one. The applications, or procedures, benefit from the service granted by lower layers, mostly the Transport Layer that gives reliable or unreliable pipes to other applications. The communications partners are distinguished by the application model, like the client-server architecture and peer-to-peer networks. This is the layer where every high level protocol, like the SMTP, FTP, SSH, HTTP, operates. Applications are addressed by ports that basically reflect services.
- **The Transport Layer:** this layer is responsible for establishing host-to-host connections on either the same or another host and on either the local networks or remote ones separated with routers. It gives a channel for the connection requirements of the processes. UDP is the main protocol for this layer, which provides unreliable datagram services. The TCP offers flowing-control, connection establishing, and reliable transmission of data.
- **The Internet Layer:** this layer performs the exchange of data-grams over the net-work boundaries. It gives a uniform net-working interface which hides the actual topology (layout) of the underlying connections. Thus, it is known as the layer establishing internetworking as well, however, it determines and establishes the Internet. This layer identifies the addressing and routing models that are utilized for the TCP/IP protocol suite. The main protocol in this domain is the Internet Protocol that defines the IP address. It has the task of transporting data-grams to the following IP router which has the connectivity to a net-work nearer to the ultimate destination of data.
- **The Link Layer:** this layer is responsible for the definition of the net-working approaches within the realm of the local net-work link where hosts communicate with no intervening routers. This layer includes the protocols which are utilized for describing the local network topology and the interfaces required for affecting transmission of Internet layer datagrams to next-neighbor hosts [48].

3.11. Client -Server Network Communication System

This communication system is done with the use of java programming language and utilized for the detection of the net-work property and utilize java serializable interface approach for storing the messages and the valuable data of the system. And this system uses the java communication packages to detect net-work card and devices and ports in addition to other undocumented hardware and software. Inter-process communication between processes on the same device or on different devices via TCP/IP sockets is a mode of point-to-point asynchronous connection. The reliability of transmissions like those is assured via the TCP protocol. However, it’s possible simulating the broadcast to a set of processes via point-to-point communication to each receiver. The roles of various applications that communicate in an application are not symmetrical, as a general rule. That description holds for client-server models. A client is capable of connecting to a service as soon as the server is ready to accept connections. For the sake of making a connection, the client has to be aware of the IP code of the server device and the port number of the service. In the case where the client has no knowledge of the IP, it must request name/number resolution with the use of the function gethostbyname. As soon as the connection is accepted by the server, every one of the programs is capable of communicating using input-output channels via the sockets that have been created at each end. The client, itself a process, sends a request to the server. The server hosts the information relevant to the application on an oracle database. Where it sends a signature image to the server, hoping for a response. A server is a process (or more than one process) accepting requests and trying to respond to them, the server receives the signature image and sends information about that signature to the client. When those programs are operated, a client and a server procedures are produced at the same time and those 2 procedures communicate with one another via reading from and writing into sockets as illustrated in Fig. (3.28).

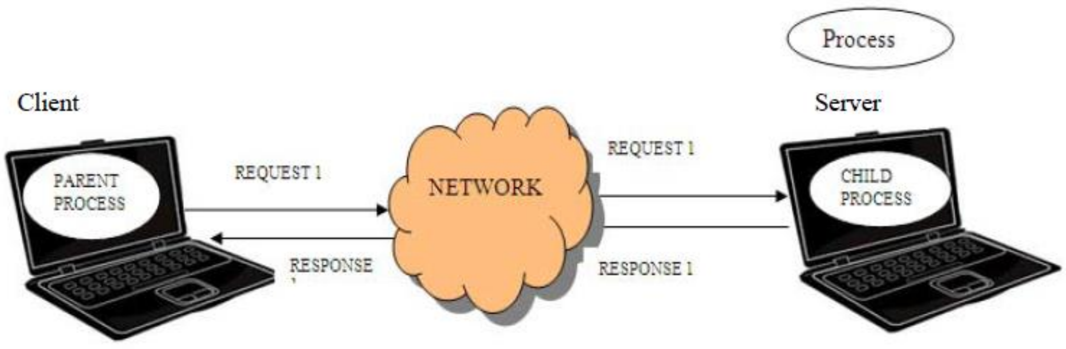


Figure 3.28 client - server communication

The server is a program receiving a request, performs the needed services and sends back the results in a reply. A server is typically capable of dealing with several requests from same or various clients simultaneously. The client and server are now capable of communicating through writing to or reading from their sockets. And once the communication is over between the client and the server, the close approach is called from each of the client and the server to close the connection as illustrated in Fig. (3.29):

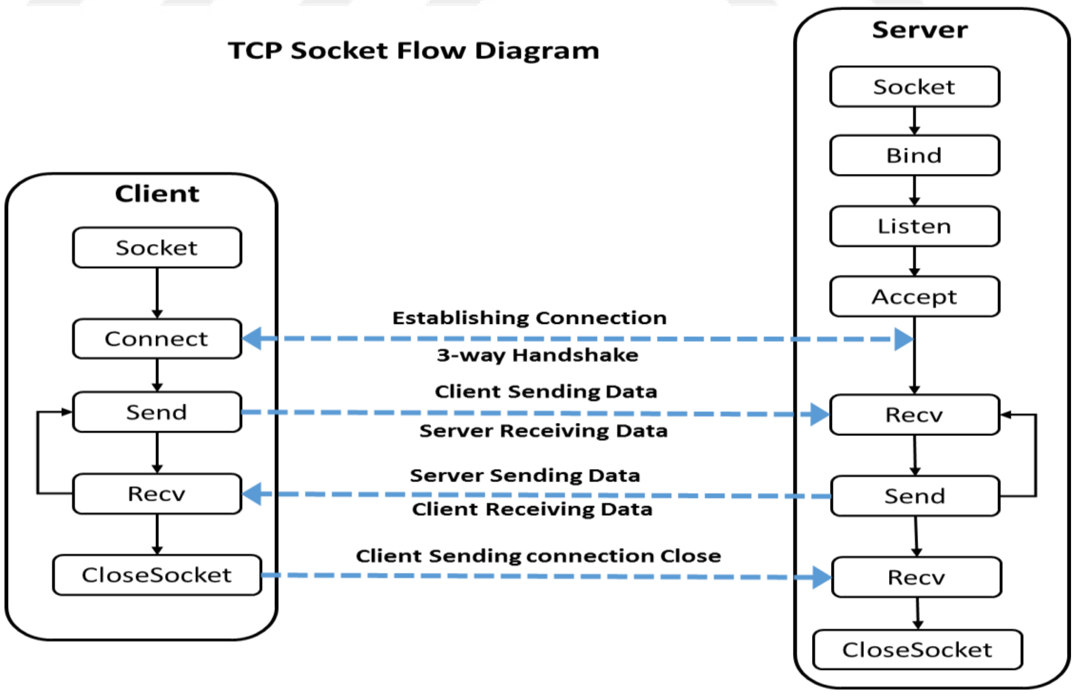


Figure 3.29 Socket based TCP Server-Client

Most commonly used socket function calls are:

Socket (): it is a data structure utilized by the socket API. When the user calls this function, it produces a socket and returns reference a socket number. That reference number will be utilized in upcoming Calls.

Bind (): This call gives the users the capability of associating a socket with a specific local port and IP address. Usually, this call is utilized on the server end. When called in server, it permits the user specifying what port and IP address arriving connections have to be addressed to. For departing connection requests, it gives the users the ability of specifying which of ports the connection will come from when checked by the other host.

Listen (): This call is responsible for preparing the socket for accepting arriving TCP requests. It has to be called prior to accept ().

Accept (): This call is responsible for detecting the arriving connection requests on the listening socket. This function will cause a task to wait until a connection request has been received.

Connect (): When a user starts this request, the stack initiates a connection with some other host. Before connect give an instruction to the stack to start a connection, the user has to pass a socket and a sockaddr_in structure that contains the IP address of the destination and its port. In TCP, the hand-shaking packets will be exchanged. Nevertheless, in UDP no packets are exchanged.

Send (): This function gives the user the ability of sending data via a connected socket. Due to the fact that the socket is already connected, it isn’t necessary specifying the address of the destination (which has been set in accept () or connect ()). The send () may be utilized for either UDP or TCP data.

Send to (): differing from send (), send to () needs the users to determine the port and address of the destination. Which is beneficial for UDP connections alone, as TCP needs a preexisting connection. The send to () can be utilized on either connected or unconnected UDP sockets. If a UDP socket is already connected, the destination address that has been given to send to () will over-ride the default established on the socket with Connect ().

Recv (): This call gives the user the ability of receiving data on the connected socket. The recv () can be utilized for either TCP or UDP.

Recvfrom (): This call gives the user the ability of receiving data from a specific UDP socket (connected or not). It may not be utilized for TCP sockets, as they need a connection.

Close (): This function is responsible for closing (deleting) a socket that is allocated with the socket call. In the case where the socket is connected, it closes the connection prior to deleting it.

3.12. Messages on Network Interface

Previous section discussed the flow of a TCP/IP application based on socket APIs. This section provide details of the messages/signals passed between two network nodes during socket function calls and the state transition of sockets on client side and server side In the following sequence diagrams, client and server implementation is represented as a combination of application and socket layers. The application layer implements the application protocol for network communication. The socket layer handles the signaling and message transfer of TCP/IP protocol using socket APIs.

3.12.1. TCP/IP Socket Opening Sequence

The starting point of socket based network application is opening socket handlers on client and server side. Fig 3.30 shows socket opening:

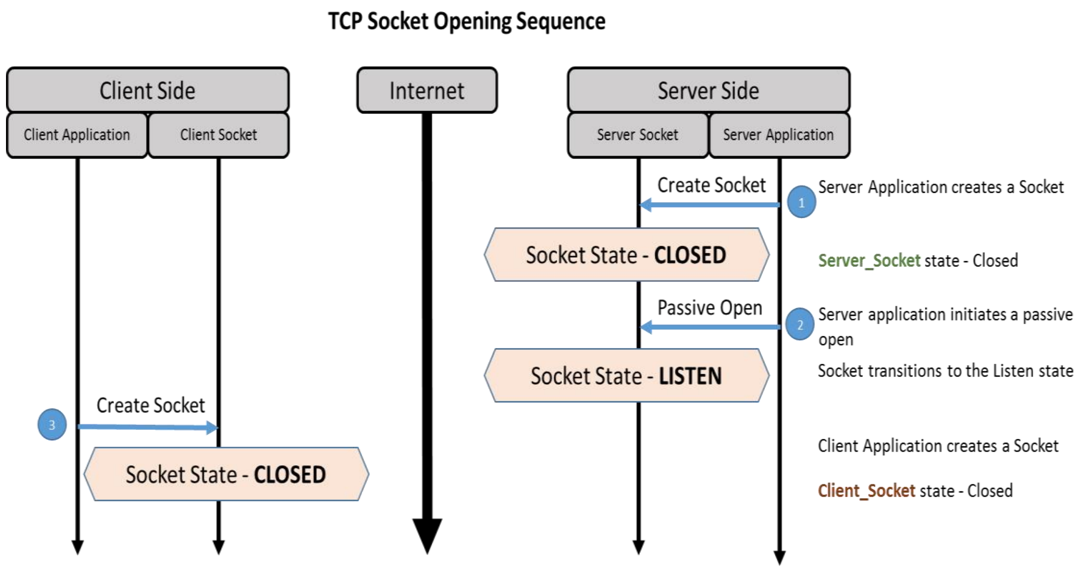

TCP Socket Opening Sequence
Client Side
Client Application
Client Socket
Internet
Server Side
Server Socket
Server Application
Create Socket
1
Server Application creates a Socket
Socket State - CLOSED
Server_Socket state - Closed
Passive Open
2
Server application initiates a passive open
Socket transitions to the Listen state
Socket State - LISTEN
Create Socket
3
Client Application creates a Socket
Socket State - CLOSED
Client_Socket state - Closed

Figure 3.30 Socket Opening.

After the initial state of socket creation is ‘Closed’ state. Then at server side, the socket will bind to a port and an IP address. After binding the socket, the socket state is transitioned to 'Listen’.

3.12.2 TCP/IP Socket Connection Sequence

The client side socket will send out active open signal when socket *connect ()* function is called. This will trigger transmission of SYN signal from client to server through physical network connection. Fig 3.31 shows Socket - Establishing Connection:

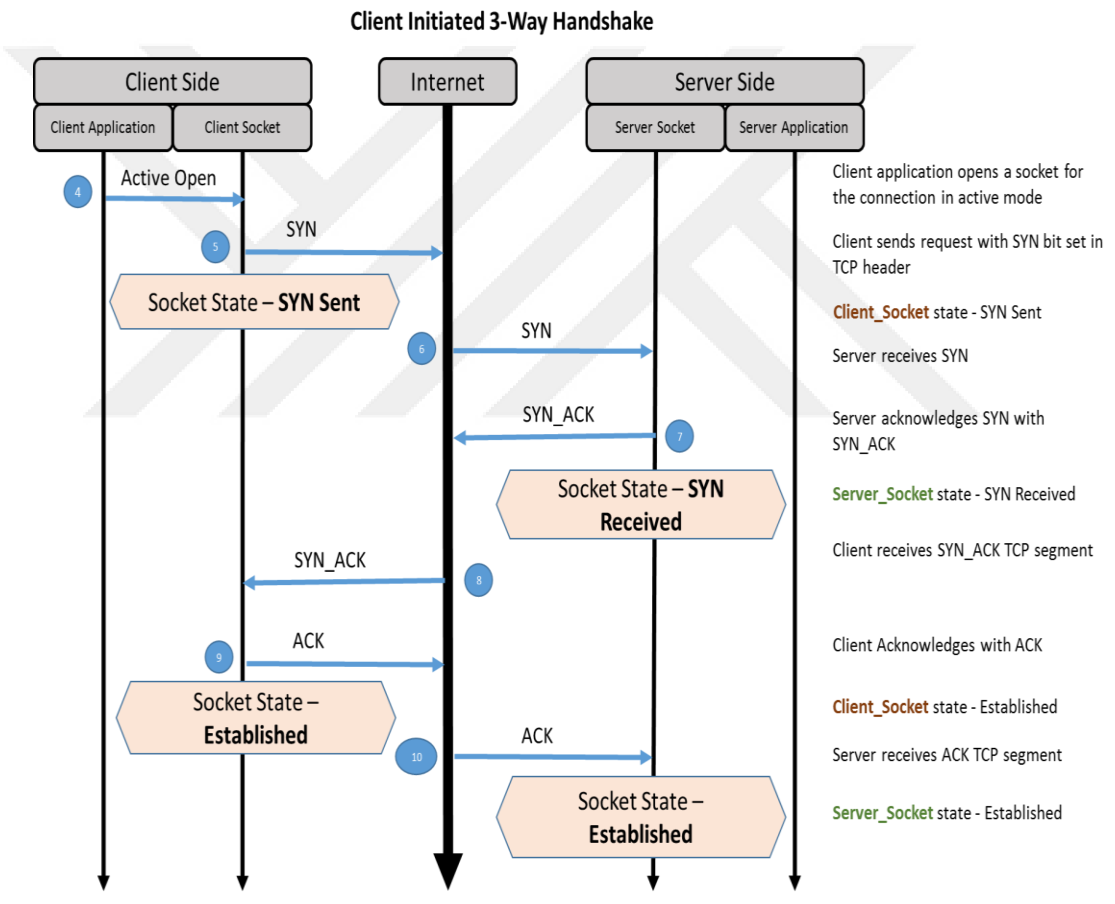

Client Initiated 3-Way Handshake
Client Side
Internet
Server Side
Client Application
Client Socket
Server Socket
Server Application
4
Active Open
5
SYN
Socket State – SYN Sent
6
SYN
SYN_ACK
7
Socket State – SYN Received
SYN_ACK
8
9
ACK
Socket State – Established
10
ACK
Socket State – Established
Client application opens a socket for the connection in active mode
Client sends request with SYN bit set in TCP header
Client_Socket state - SYN Sent
Server receives SYN
Server acknowledges SYN with SYN_ACK
Server_Socket state - SYN Received
Client receives SYN_ACK TCP segment
Client Acknowledges with ACK
Client_Socket state - Established
Server receives ACK TCP segment
Server_Socket state - Established

Figure 3.31 Socket - Establishing Connection

The SYN will be received and acknowledged by server. Then server will send its SYN signal, together with acknowledgment to the client. On receipt of acknowledgment from Server, the socket at client side will transition to ‘Established’ state. The client will acknowledge the SYN message from server. On receipt of acknowledgment from Client, the socket at server side will transition to ‘Established’ state.

3.12.3.TCP/IP Socket Data Transfer Sequence between Client and Server.

When the connection is established, the data can be transferred between server and client. The following diagram as shown below for the typical data transfer between a server and client. The client application send a data packet to server. If the size of packet is higher than the maximum size limit of TCP segment, the TCP layer at client will split packet into multiple TCP segments. At server side on receipt of TCP segments, the server socket acknowledges the receipt of data from client. Server is acknowledging after receipt of two TCP segments. Similarly server can also send data to client and confirm the transfer by checking acknowledgment from Client. Fig. 3.32 shows   Socket - Transferring Data

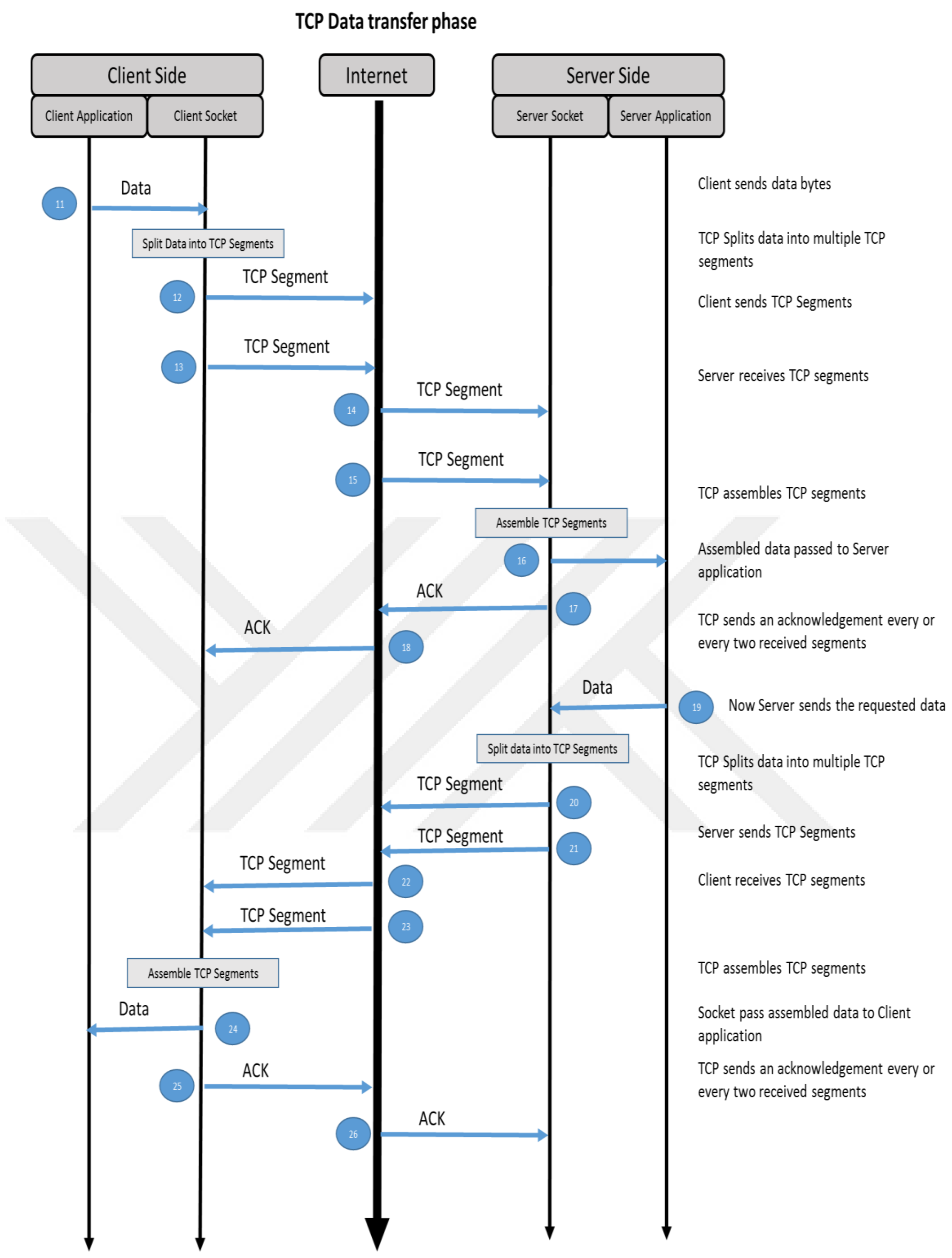

TCP Data transfer phase
Client Side
Internet
Server Side
Client Application
Client Socket
Server Socket
Server Application
11
Data
Client sends data bytes
Split Data into TCP Segments
TCP Splits data into multiple TCP segments
12
TCP Segment
Client sends TCP Segments
13
TCP Segment
14
TCP Segment
Server receives TCP segments
15
TCP Segment
TCP assembles TCP segments
Assemble TCP Segments
16
Assembled data passed to Server application
ACK
17
ACK
18
TCP sends an acknowledgement every or every two received segments
Data
19
Now Server sends the requested data
Split data into TCP Segments
TCP Splits data into multiple TCP segments
TCP Segment
20
TCP Segment
21
Server sends TCP Segments
TCP Segment
22
Client receives TCP segments
TCP Segment
23
Assemble TCP Segments
TCP assembles TCP segments
Data
24
Socket pass assembled data to Client application
ACK
25
TCP sends an acknowledgement every or every two received segments
ACK
26

Figure 3.32 Socket - Transferring Data

3.12.4.TCP/IP Socket Closing Sequence

When the data transfer between server and client is complete, the communication sockets will be closed. On calling socket *close ()* function by client application, client will send FIN message to server. On receipt of FIN message by server socket, it acknowledges and transition the server socket to ‘close wait’ state. When client receives the acknowledgment, it changes the state of client socket to ‘FIN Wait 2’. When server completes the data transfer, it calls socket *close ()*. Then FIN message will be send to client. On receipt of FIN message by client socket, it acknowledges and transition the client socket to ‘Time wait’ state. A timer will be started at client side. When timer expires, the client socket state will change to ‘Closed’ and releases the resources for client socket. When server receive the acknowledgment, it changes the state of client to ‘Closed’ and releases the resources used by server socket. Fig 3. 33 shows Socket - Closing Connection.

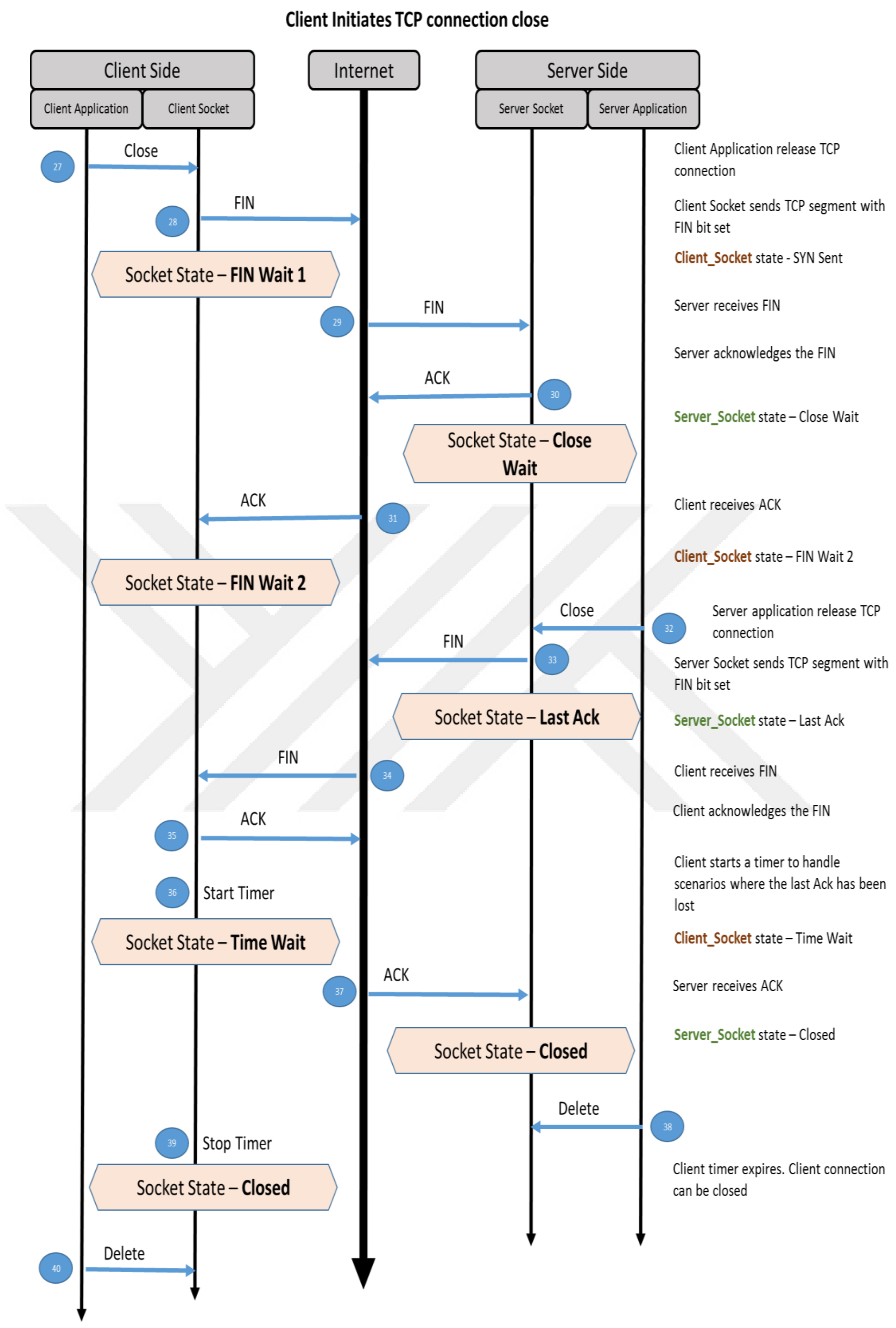

Client Initiates TCP connection close
Client Side
Internet
Server Side
Client Application
Client Socket
Server Socket
Server Application
27
Close
Client Application release TCP connection
28
FIN
Client Socket sends TCP segment with FIN bit set
Socket State – FIN Wait 1
Client_Socket state - SYN Sent
29
FIN
Server receives FIN
ACK
30
Server acknowledges the FIN
Socket State – Close Wait
Server_Socket state – Close Wait
ACK
31
Client receives ACK
Socket State – FIN Wait 2
Client_Socket state – FIN Wait 2
Close
32
Server application release TCP connection
FIN
33
Server Socket sends TCP segment with FIN bit set
Socket State – Last Ack
Server_Socket state – Last Ack
FIN
34
Client receives FIN
35
ACK
Client acknowledges the FIN
36
Start Timer
Client starts a timer to handle scenarios where the last Ack has been lost
Socket State – Time Wait
Client_Socket state – Time Wait
37
ACK
Server receives ACK
Socket State – Closed
Server_Socket state – Closed
Delete
38
39
Stop Timer
Client timer expires. Client connection can be closed
Socket State – Closed
40
Delete

Figure 3.34 Socket - Closing Connection.

CHAPTER 4

EXPERIMENTAL RESULTS

4.1. Dataset

There are various kinds of features for signature recognition. SURF and SIFT features have been used in the off-line signature recognition. A total of 10 persons signatures have been used image in the experimentation. Every person includes 24 signature composed of 160 training signatures and 80 testing signatures and the total signature image 240 signatures the images are converted in to gray scale. Each of the pictures has been stored in PNG format in database. . Fig. (4.1). shows the dataset of this experiment.

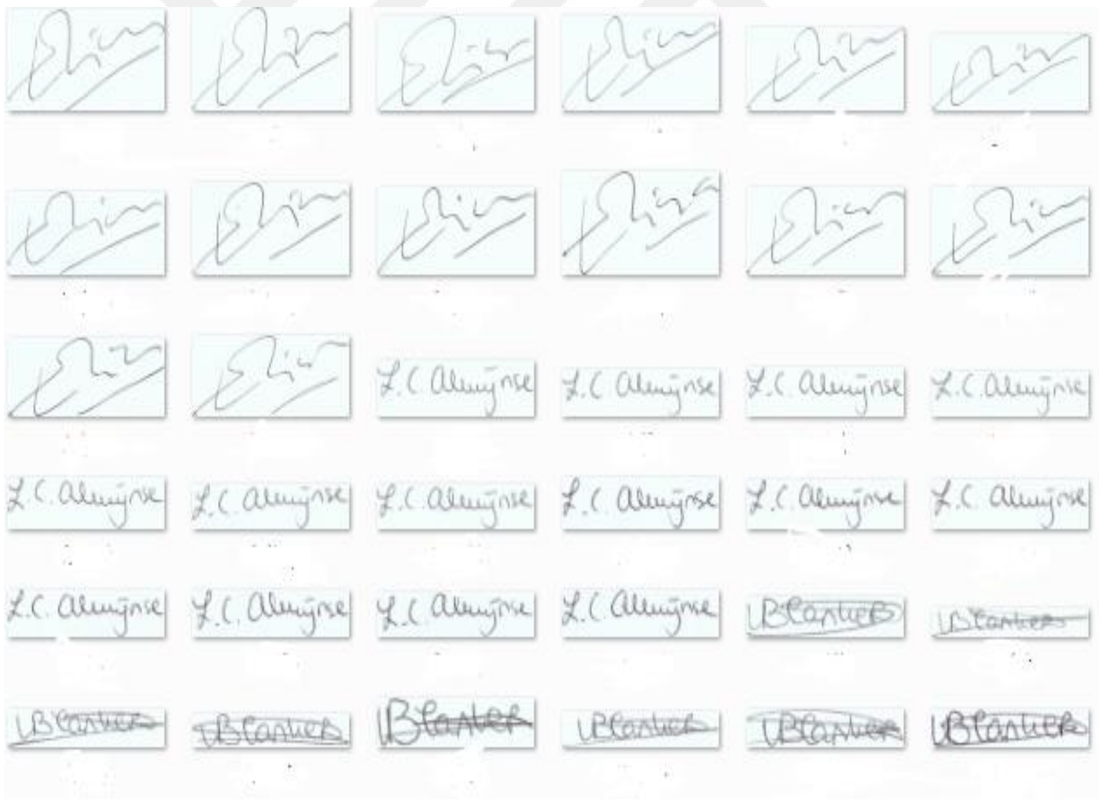


Figure 4.1 Samples of dataset.

4.2. Feature Detection Results

While applying the property detection component on the images it has been found that despite the image, SIFT finds more properties than SURF. Taking under consideration that SIFT is an established robust property detector, SIFT and SURF have been implemented on the images and the following results (see Fig.4.2 and 4.3) can illustrate that those both approaches are capable of recognizing signatures images under different situations.

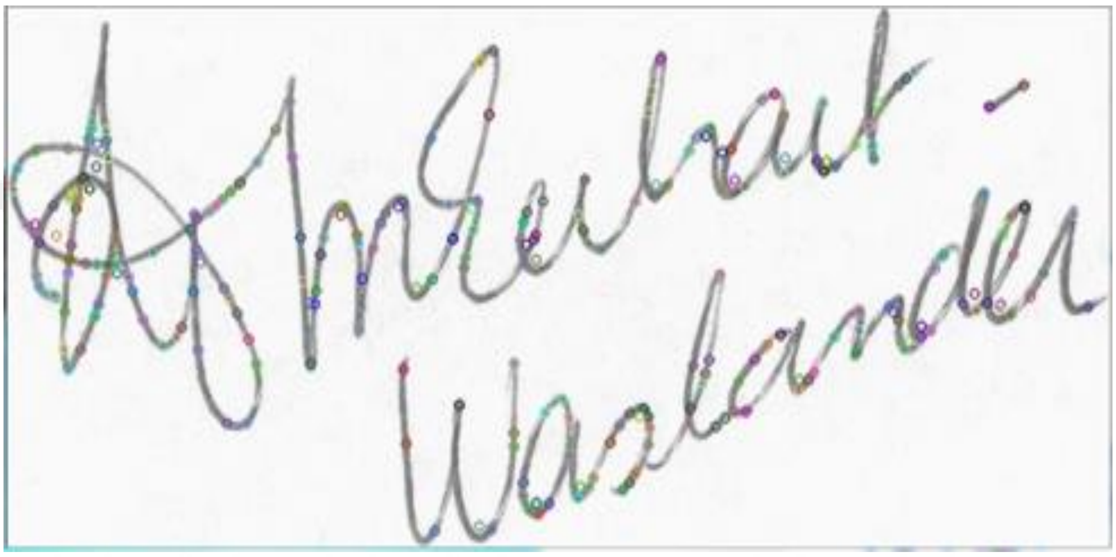


Figure 4.2 Key point detector by using SIFT algorithm.

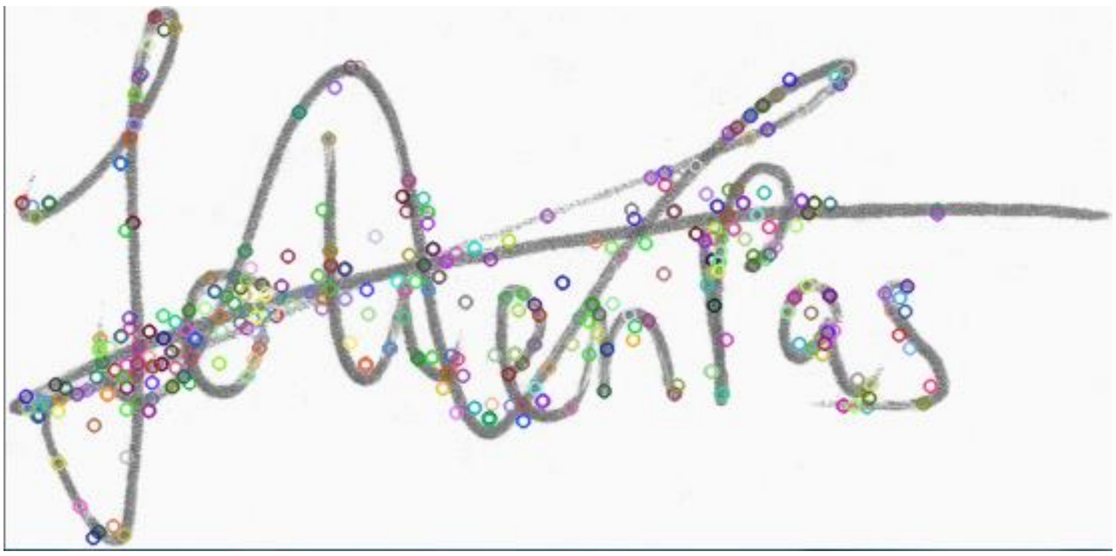


Figure 4.3 Key point detector by using SURF algorithm**.**

4.3. Comparing between SIFT and SURF

SIFT and SURF have been applied on the given signatures images in the data-base. Table 1 lists the number of properties that have been detected with the use of SIFT and SURF implemented on six signatures 1, 2, 3, 4, 5 and 6 obtained from the data-base. As seen in the table that the number of the properties that have been obtained from SIFT is more than the features obtained from the SURF. Table 2 lists a detailed comparing of the classification method results based on SIFT and SURF with support vector machine based on RBF kernel.

Table 4.1 Table listing the number of properties using SIFT and SURF

NUMBER OF FEATURES			
S .No.	**Signature image**	**SIFT FEATURES**	**SURF FEATURE**
1	1	400	344
2	2	381	330
3	3	363	248
4	4	354	309
5	5	398	299
6	6	400	355

Table 4.2 The Compression of the experimental result.

Algorithm	**Correctly classified images**	**Average accuracy**	**Time of Training (seconds)**	**Time of Testing (seconds)**
SVM classifier based on RBF kernel with SIFT	79/80	98.75%	0.598	0.911
SVM classifier based on RBF kernel with SURF	78/80	97.5%	0.013	0.597

The following plot graph shows the comparison between those two algorithms.

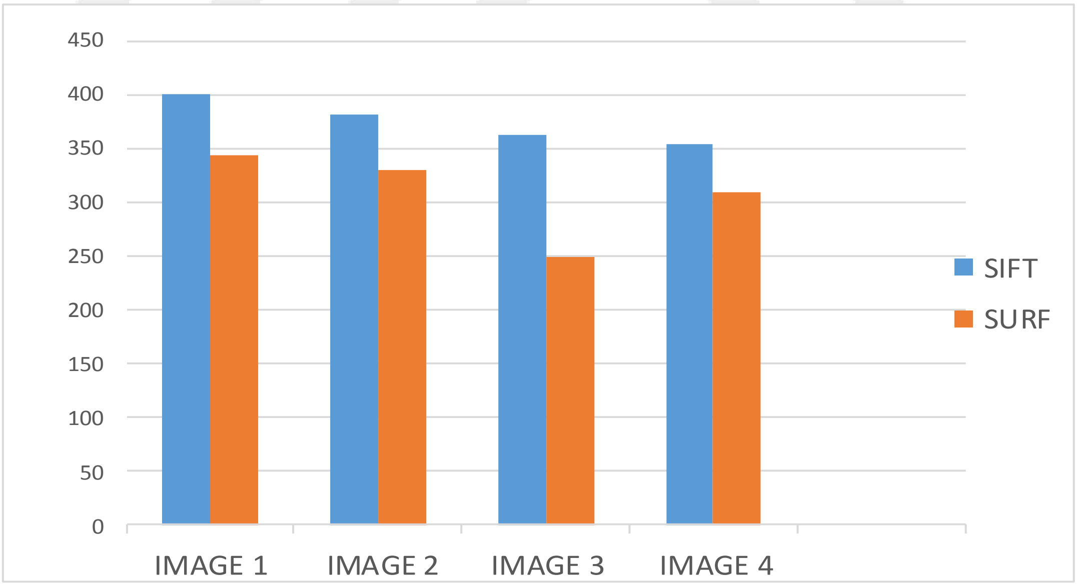
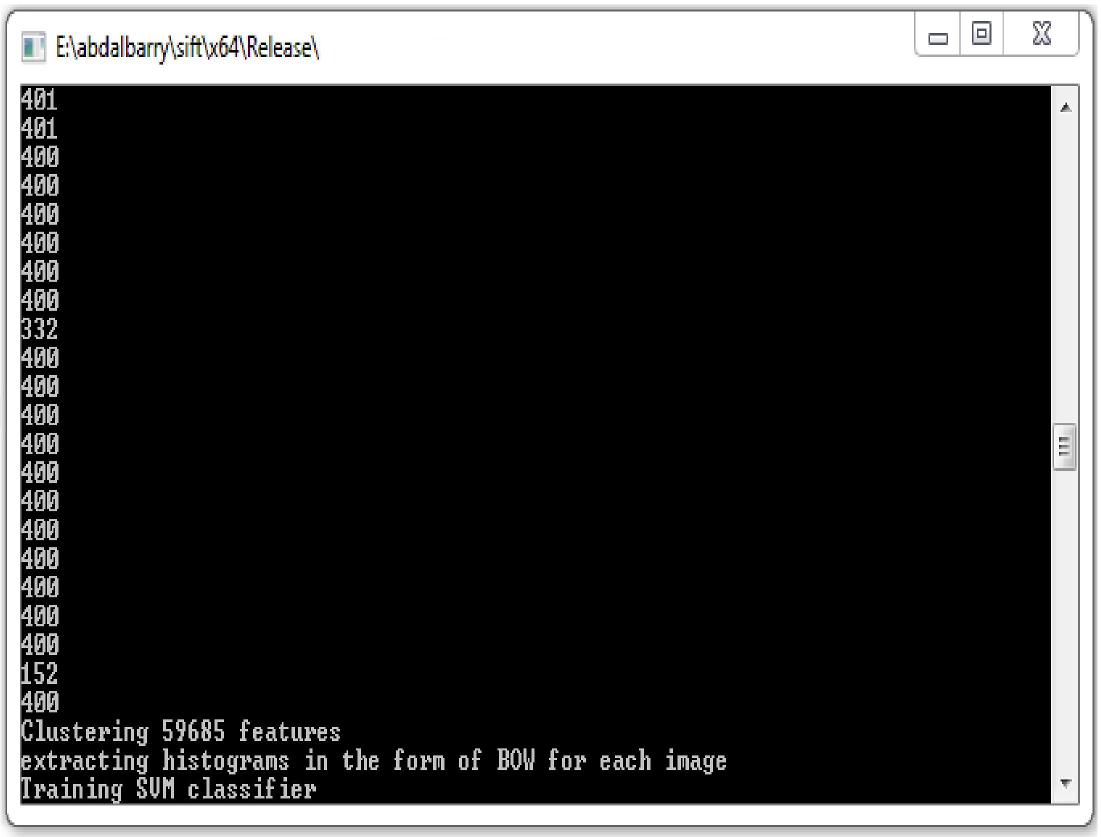


Figure 4.4 Graph comparing the number of features found with the use of SIFT and SURF number for signature images

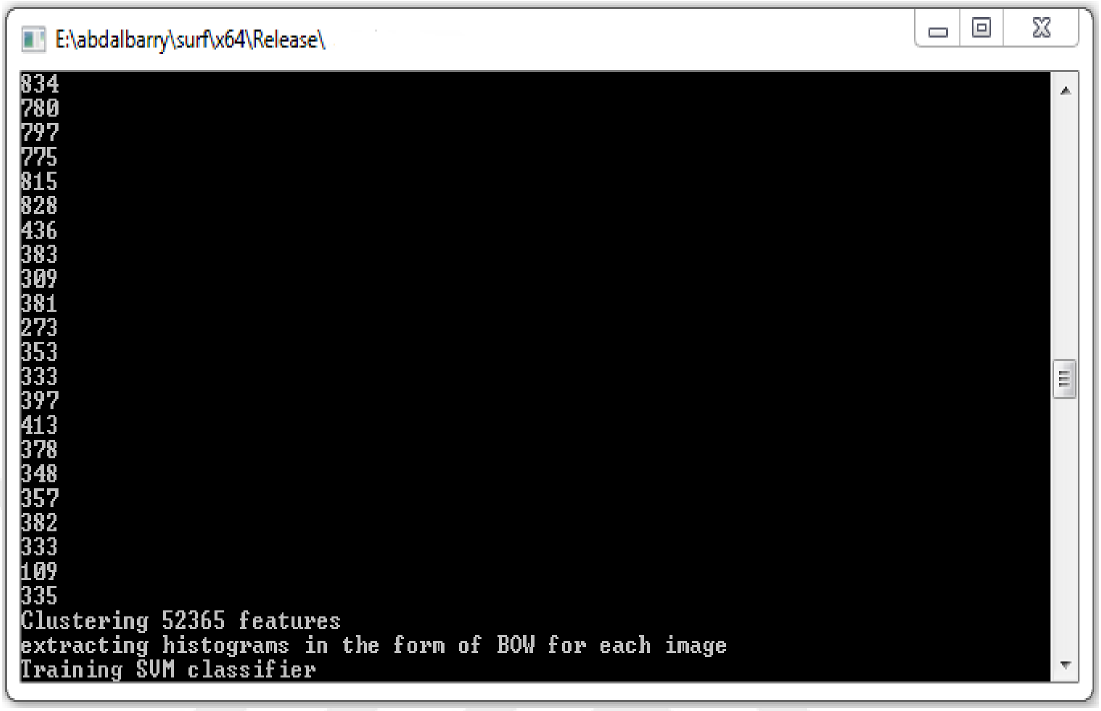
As it is noted from Fig. (4.5), (4.6) the disadvantage of complex computation has been solved using this approach. In addition, more global information has been able to be reserved, which can ensure a better image classification precision.



```
401
401
400
400
400
400
400
400
332
400
400
400
400
400
400
400
400
400
400
400
152
400
Clustering 59685 features
extracting histograms in the form of BOW for each image
Training SVM classifier
```

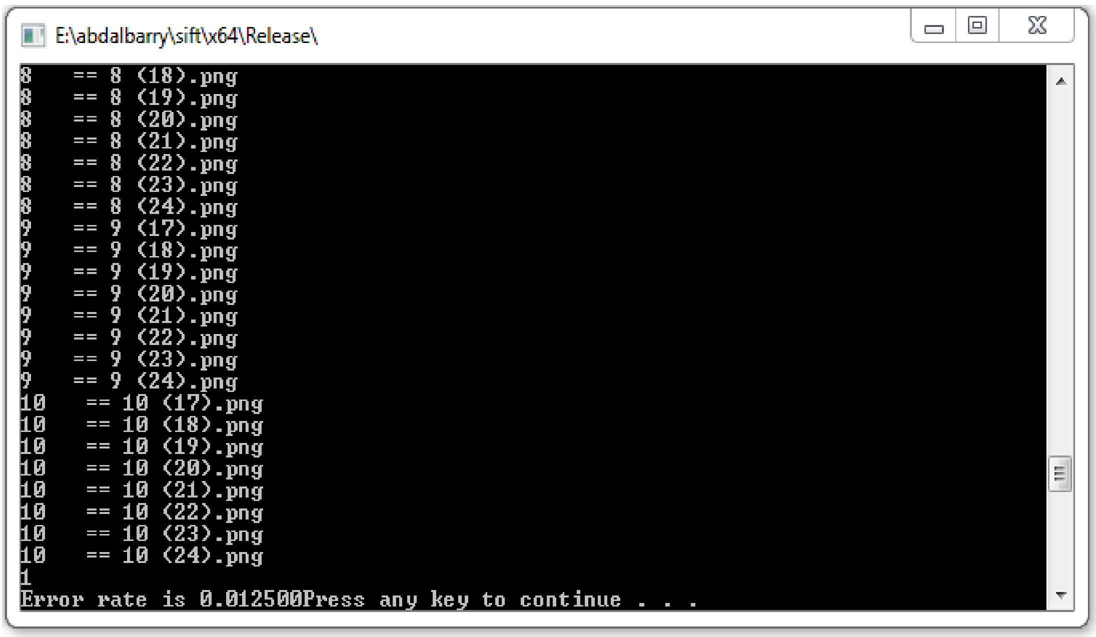
Figure 4.5 Results of Classifying with the Use of Library Size of 240 signature Images for SIFT

Dataset	Time training data set (h:m:s)	Time training data set (h:m:s)
Sample 1		
Sample 2		
Sample 3		
Sample 4		
Sample 5		
Sample 6		



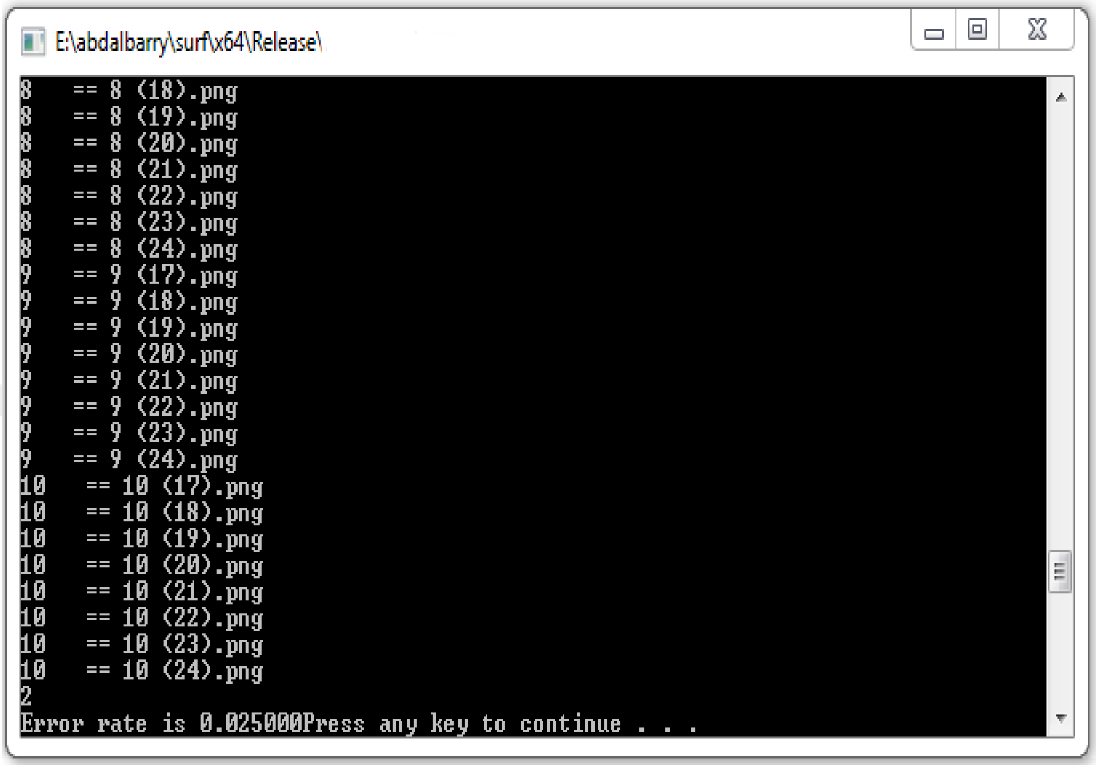
```
834
780
797
775
815
828
436
383
309
381
273
353
333
397
413
378
348
357
382
333
109
335
Clustering 52365 features
extracting histograms in the form of BOW for each image
Training SVM classifier
```

Figure 4.6 Results of Classifying with the Use of Library Size of 240 signature Images for SURF



```
8    == 8 (18).png
8    == 8 (19).png
8    == 8 (20).png
8    == 8 (21).png
8    == 8 (22).png
8    == 8 (23).png
8    == 8 (24).png
9    == 9 (17).png
9    == 9 (18).png
9    == 9 (19).png
9    == 9 (20).png
9    == 9 (21).png
9    == 9 (22).png
9    == 9 (23).png
9    == 9 (24).png
10    == 10 (17).png
10    == 10 (18).png
10    == 10 (19).png
10    == 10 (20).png
10    == 10 (21).png
10    == 10 (22).png
10    == 10 (23).png
10    == 10 (24).png
1
Error rate is 0.012500Press any key to continue . . .
```

Figure 4.7 Error rate by using SIFT with SVM basic on RBF kernel



```
8    == 8 (18).png
8    == 8 (19).png
8    == 8 (20).png
8    == 8 (21).png
8    == 8 (22).png
8    == 8 (23).png
8    == 8 (24).png
9    == 9 (17).png
9    == 9 (18).png
9    == 9 (19).png
9    == 9 (20).png
9    == 9 (21).png
9    == 9 (22).png
9    == 9 (23).png
9    == 9 (24).png
10    == 10 (17).png
10    == 10 (18).png
10    == 10 (19).png
10    == 10 (20).png
10    == 10 (21).png
10    == 10 (22).png
10    == 10 (23).png
10    == 10 (24).png
2
Error rate is 0.025000Press any key to continue . . .
```

Figure 4.8 Error rate by using SURF with SVM basic on RBF kernel

Accuracy of SIFT=$\frac{79}{80}\times 100$=98.75%

Accuracy of SURF=$\frac{78}{80}\times 100$ =97.5%

Classification accuracy= Number of recognized signatures /Total number of testing signatures

4.4. Data Transfer between a Server and Client

The client/server network communication system can be started via launching the Data Server. Java program is illustrated in Fig (4.9).

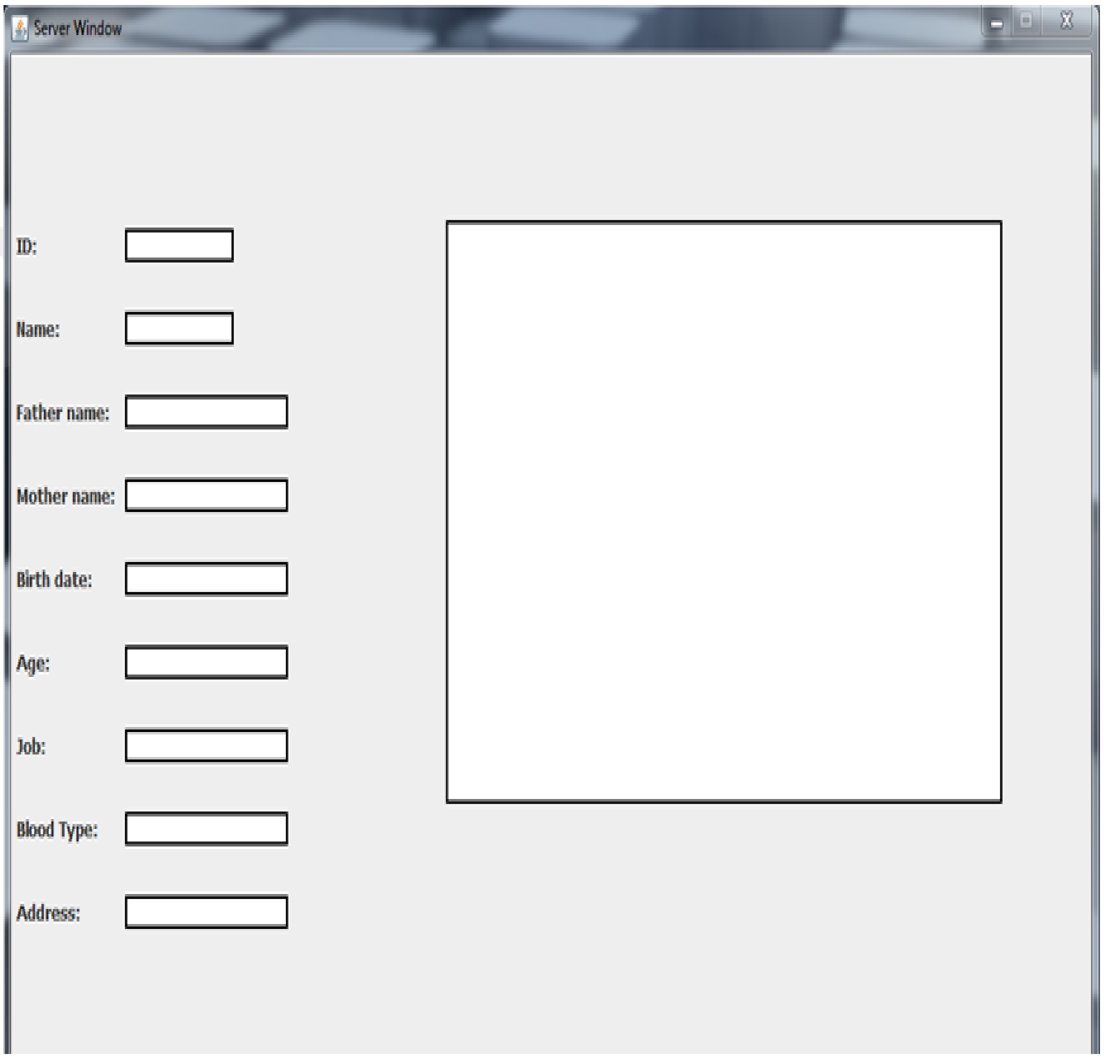


Figure 4.9 Starting the Server Design

When a user requests the initiation of the client program, they have to execute that program. It will start as illustrated in Fig. (4.10).

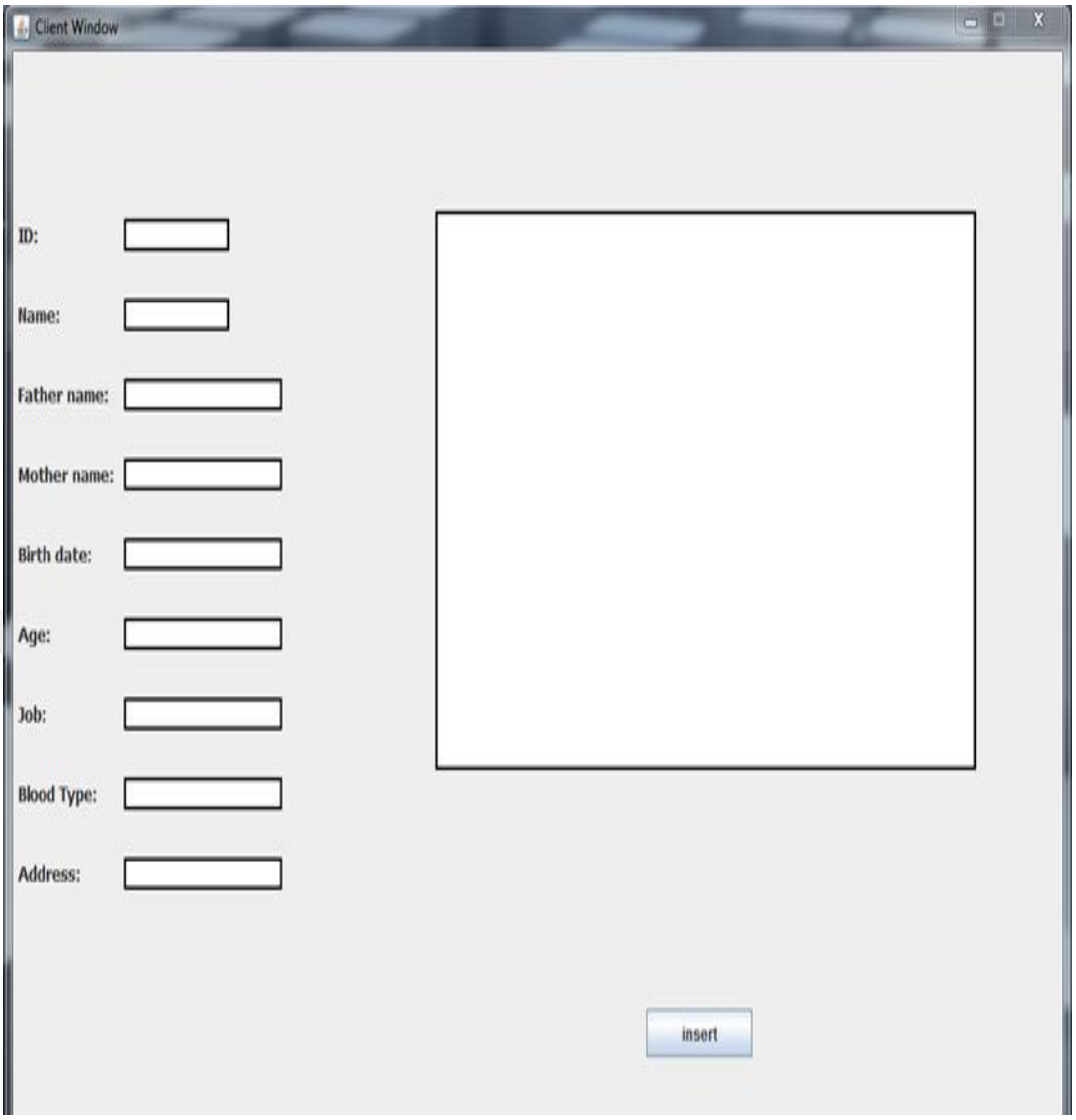


Figure 4.10 Starting the Client Design

When the connection is configuration, the data can be transferred between server and client. The client send signature image to server, at server side on receipt of TCP segments, the server socket acknowledges the receipt of data from client. Similarly server work on send signature image to client and include information about person signature image. In Fig (4.11) the user selects the signature image and sends it to the server.

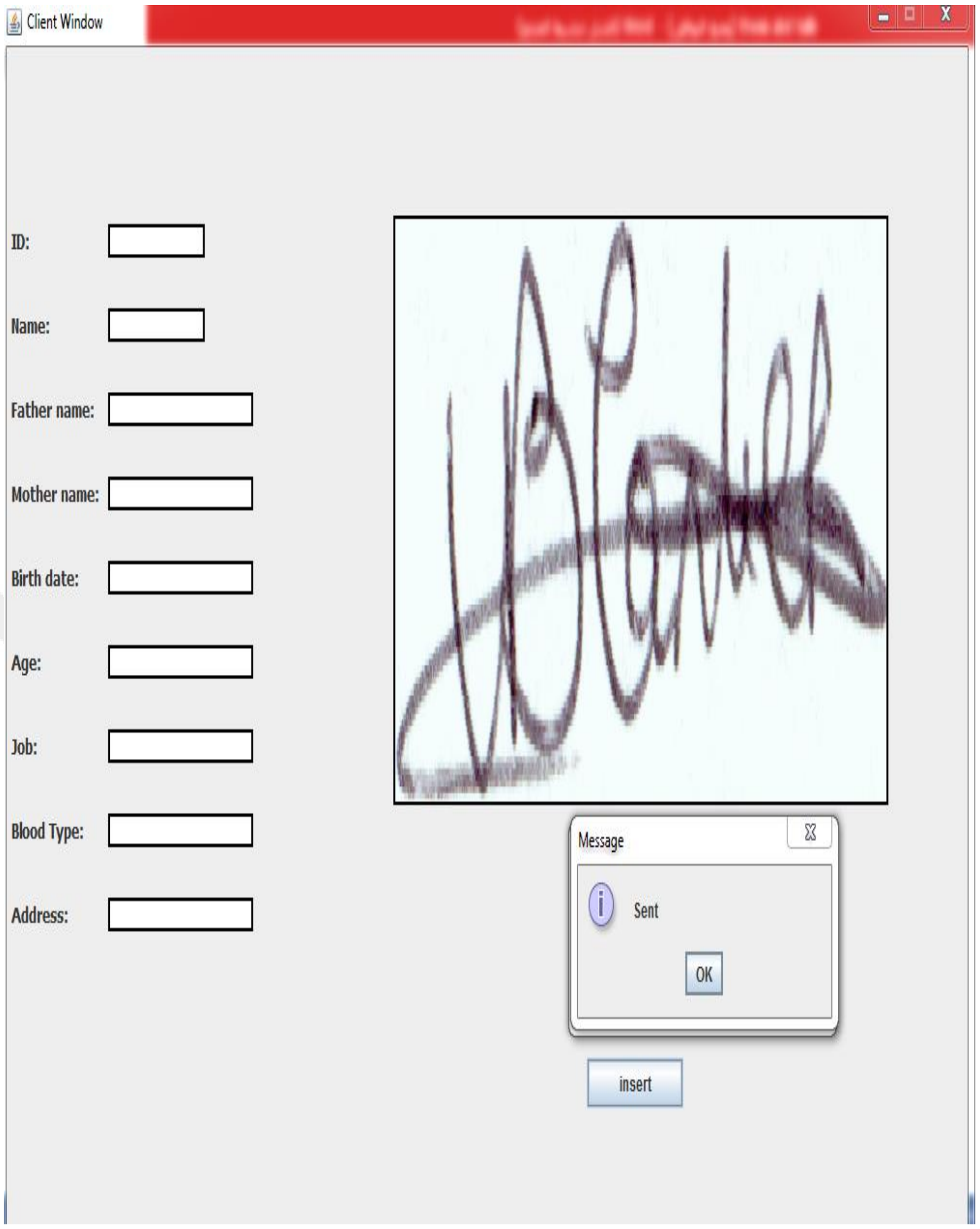
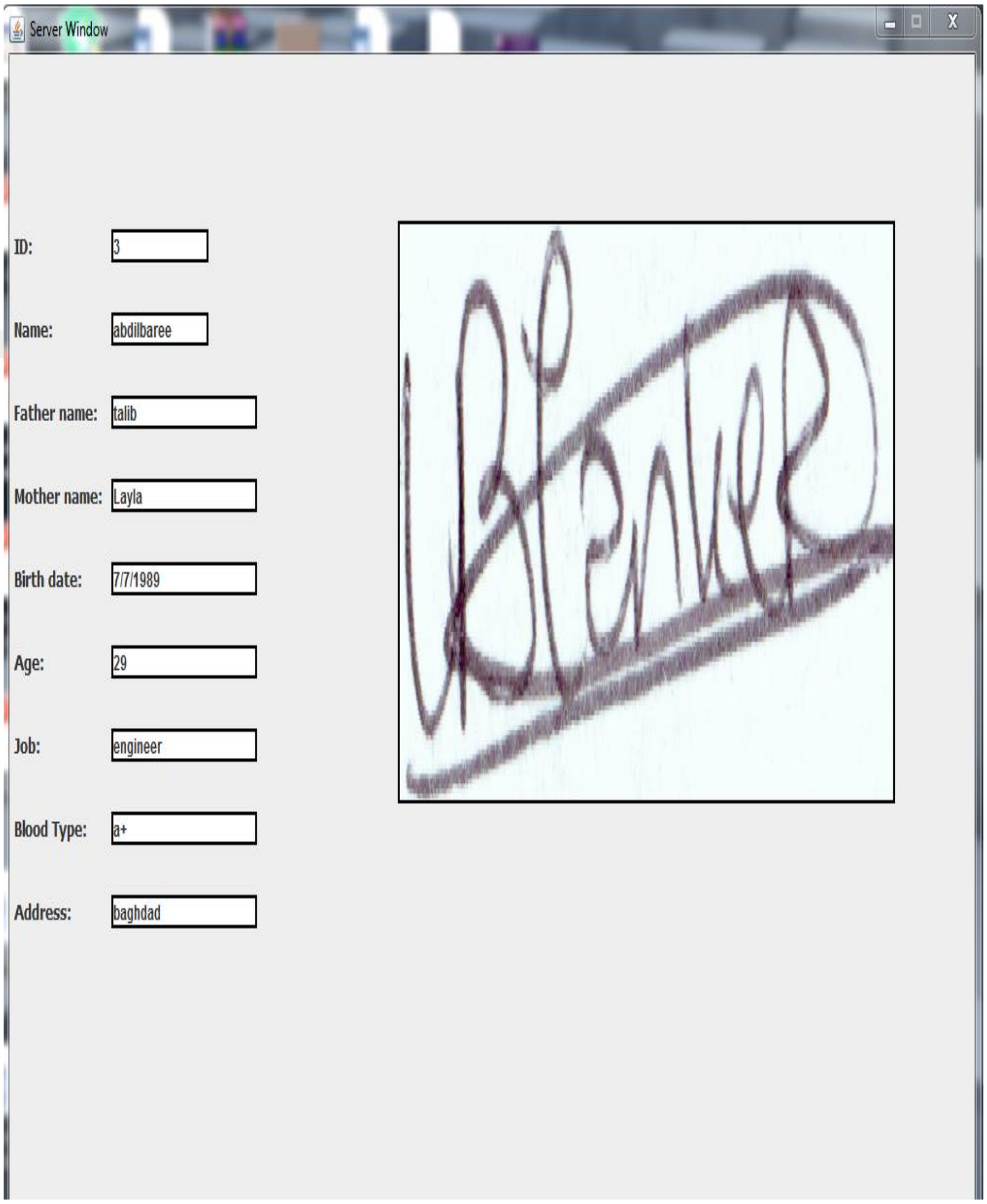


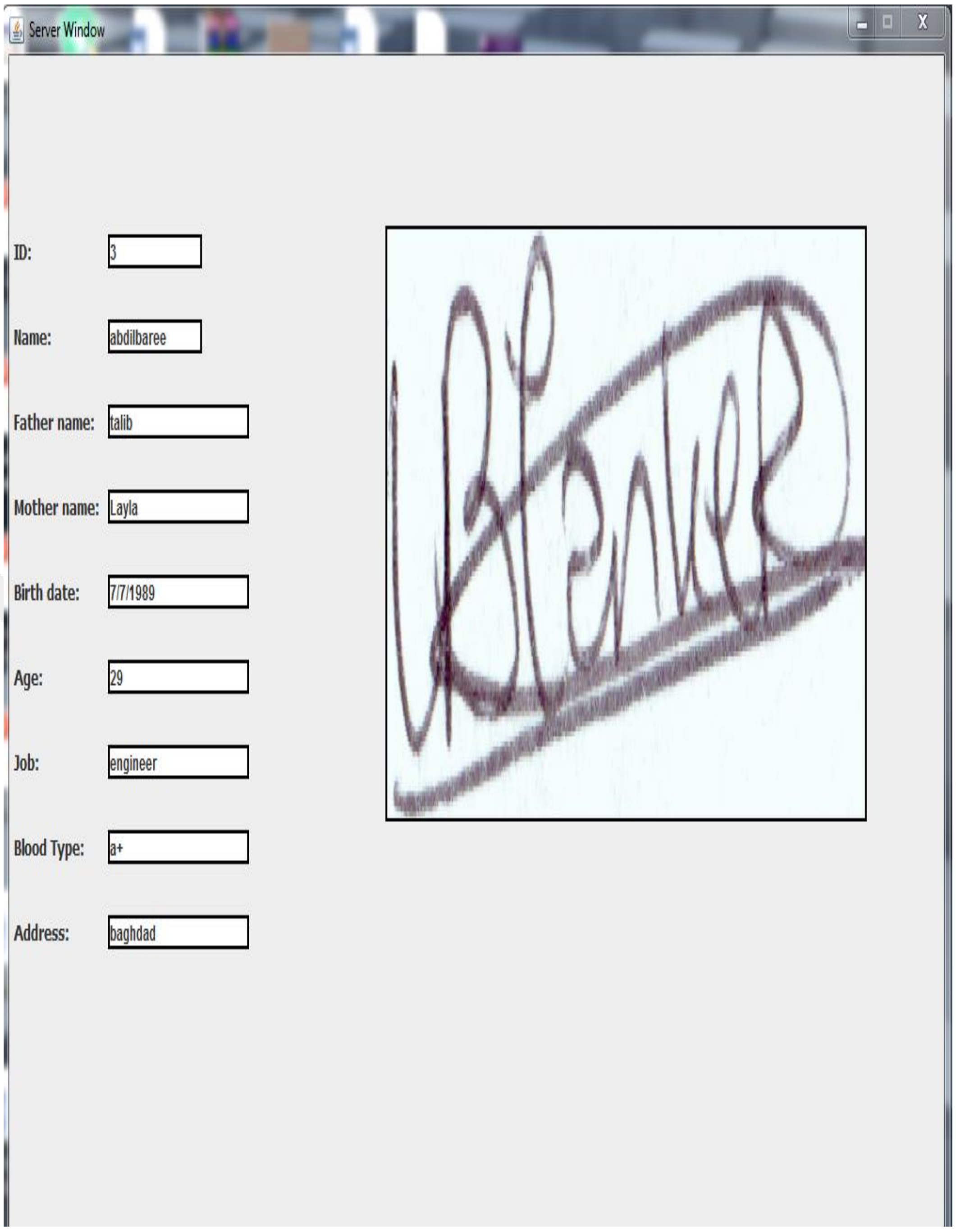
Figure 4.11 Send data (signature image) from client to server

Server receives signature image from client and it processes and executes that command and gives information about person signature image, as shown in Fig (4.12). And in the same time the requested data is sent to the client by the server as shown in Fig (4.13).



ID:	3
Name:	abdilbaree
Father name:	talib
Mother name:	Layla
Birth date:	7/7/1989
Age:	29
Job:	engineer
Blood Type:	a+
Address:	baghdad

Figure 4.12 Server receives data (signature image) and give information about person signature image



ID:	3
Name:	abdilbaree
Father name:	talib
Mother name:	Layla
Birth date:	7/7/1989
Age:	29
Job:	engineer
Blood Type:	a+
Address:	baghdad

Figure 4.13 Client received data about person signature image from server

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

5.1 Conclusion

In this study, an off-line signature identification with the use of SIFT and SURF property extraction with SVM basic of radial basis function was proposed., These can be experimental results in the system showing that when we use algorithm SVM basic on RBF performance, it is better than algorithm SURF with SVM basic on RBF. There are some applications that use a SIFT and SURF detectors for signature image detection. Nevertheless, the results of experimentations on signature image detection proves that the SIFT detector works more efficiently than with the SURF detector. The accuracy of signature image detection algorithm is various according to the parameters of the SIFT key-point detector. The use of SVM can make signature recognition system more efficient as compared with other existing systems according to the success ratio and ease of implementation and optimized run time. This system may provide a solution in facilitating the remote election process. In addition there is a description of the socket programming in java over TCP. It is found that the use of SIFT with SVM-RBF kernel system, has an accuracy of 98.75% and SURF with SVM-RBF kernel has an accuracy of 97.5%.and the error rate for SIFT with SVM-RBF kernel function is 0.0125 and error rate of SURF with SVM-RBF kernel function is 0.025.

5.2 Future Works

This thesis or work is limited to acquiring signature image recognition from data base and how to send and receive data between client and server. For future work, it is possible to use different image based biometrics such as face, fingerprint, and iris recognition for this system. and can we use this method or this algorithms SIFT and SURF with SVM which we applied in my research work is also applied on dynamic images, and after getting the result we should compare the result with the present research work.

REFERENCES

Ammar, M. (1990). Performance of parametric and reference pattern based features in\nstatic signature verification: a comparative study Proceedings, 10th Int. Conf. Pattern Recognit., 1, 646–648.

Hess, R. (2010).An open-source SIFTLibrary, Int. Conf. Multimed., 1493–1496. Cortes, C., Vapnik, V. (1995). Support Vector Networks, Mach. Learn., 20, 273-297.

Jadhav, N. V. Panchal, S. C., Rotti, M. R. (2013). Client Server Network Management System for WLAN (Wi-Fi) with Remote Monitoring, Int. J. Sci. Res. Netw. Secur. Commun, 1, 22–25.

Al-shoshan, A. I., Arabia, S. (2006). Handwritten Signature Verification Using Image Invariants and Dynamic Features, Int. Conf. Comput. Graph. Imaging Vis., 173–176.

Abdelrahman, A., Abdallah, A. (2013) Signature Verification System Based on Support Vector Machine Classifier.

Özgündüz, E., Şentürk, T., Karslıgil, M. E. (2005). Off-line Signature Verification and Recognition by Support Vector Machine, Eur. Signal Process, 90, 1–4.

Shahri, R. P., Pourreza, H. (2009). Offline signature verification using local radon transform and support vector machines, Int. J. Image, 5, 184–194.

Sigari, M. H., Pourshahabi, M. R., Pourreza, H. R. (2000). Offline Handwritten Signature Identification using Grid Gabor Features and Support Vector Machine, 281–286.

Ruiz-del-Solar, J., Devia, C., Loncomilla, P., Concha, F. (2008). Offline Signature Verification Using Local Interest Points and Descriptors, 13th Iberoam. Congr. Pattern Recognit, 5197, 22–29.

Liu, T., Kim, S.-H., Lee, H.-S., Kim, H.-H. (2009). Face recognition base on a new design of classifier with SIFT keypoints, IEEE International Conference on Intelligent Computing and Intelligent Systems, 366–370.

Hua, S., Chen, G., Wei, H. Jiang, Q. (2012). Similarity measure for image resizing using SIFT feature, EURASIP J. Image Video Process., 2012, 6.

Yan Ke, Sukthankar, R. (2004). PCA-SIFT a more distinctive representation for local image descriptors, in Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2, 506–513.V. Pagar and P. N. G. Pardeshi.(2014). Design and Implementation Handwritten, **4**, 811–814.Ahmed, S., Malik, M. I., Liwicki, M., Dengel, A. (2012). Signature segmentation from document images, Proc. - Int. Work. Front. Handwrit. Recognition, IWFHR, 425–429.

Malik, M. I., Liwicki, M., Dengel, A., Uchida, S., Frinken, V. (2014). Automatic Signature Stability Analysis and Verification Using Local Features, International Conference on Frontiers in Handwriting Recognition, 621–626.

Bairagi, B. K., Chatterjee, A., Das, S. C., Tudu, B. (2012). Expressions invariant face recognition using SURF and Gabor features, International Conference on Emerging Applications of Information Technology, 170–173.

Lin, S., Liu, B., Lin, J. (2012). Combining Speeded-Up Robust Features With Principal Component Analysis in Face Recognition System, J. Innov. Comput. Inf, 8, 8545–8556.

Bay, H., Ess, A., Tuytelaars, T., Van Gool, L. (2008). Speeded-Up Robust Features,

Comput. Vis. Image Underst., 110, 346–359.

Huang, K., Yan, H. (1997). Off-line signature verification based on geometric feature extraction and neural network classification, Pattern Recognit., 30, 9–17.

Khuwaja, G. A., Laghari, M. S. (2011). Offline Handwritten Signature Recognition,

5, 1300–1303.

Abikoye, O. C., Mabayoje, M., Ajibade, R. (2011) “Offline Signature Recognition & Verification using Neural Network, Int. J. Comput. Appl. (0975 – 8887), 35, 44–51.

Fleuret, F., Geman, D. (1999). Graded Learning for Object Detection, Ieee Work. Stat. Comput. Theor. Visions, 544--549.

Mikolajczyk, K., Schmid, C. (2005). A Performance evaluation of local descriptors.

IEEE Trans. Pattern Anal. Mach. Intell. pattern Anal. Mach. Intell., 27, 1615–30.

Kekre, H. B., a Bharadi, V. (2010). Off-Line Signature Recognition Systems, Int. J. Comput. Appl., 1, 61–70.

Shah, V., Sanghavi, U., Shah, U. (2013). Off-line signature verification using curve fitting algorithm with neural networks, International Conference on Advances in Technology and Engineering (ICATE), 1–5.

Basavaraj, L., Samuel, R. D. S. (2009). Offline-line Signature Verification and Recognition : An Approach Based on Four Speed Stroke Angle, 2, 40–42.

Odeh, S., Khalil, M. (2011). Apply Multi-Layer Perceptrons Neural Network for Off-line signature verification and recognition, Int. J. Comput. Sci. Issues, 8, 261–266.

Raja, K. B. (2012). Off-line Signature Verification based on Angular Features, Int.

J. Model. Optim, 477–481.

Ghandali, S., Moghaddam, M. E. (2009). Off-Line Persian signature identification and verification based on image registration and fusion, J. Multimed., 4, 137–144.

Shirdhonkar, M. S., Kotare, M. B. (2011). Off-line Handwritten Signature Identification Using Rotated Complex Wavelet Filters, J. Comput. Sci., 8, 478–482.

Ho, C. W., Deng, P. S. (2005). Wavelet–based Off–line Signature Verification, 1–28.

Moghaddam, B., Pentland, A. (1997). Probabilistic visual learning for object representation, IEEE Trans. Pattern Anal. Mach. Intell., 19, 696–710.

Keypoints, S., Lowe, D. G. (2004). Distinctive Image Features from, Int. J. Comput. Vis., 60, 91–110.

Coetzer, J., Herbst, B., du Preez, J. (2004). Offline signature verification using the discrete radon transform and a hidden Markov model, EURASIP J. Appl., 559–571.

Grabner, H. M. (2011). Fast approximated SIFT, 918–927.

Silpa-Anan, C., Hartley, R. (2008). Optimised KD -trees for fast image descriptor matching, IEEE Conf. Comput. Vis. Pattern Recognit., 1–8.

Witkin, A. P. (1984). SCALE-SPACE FILTERING, 1–4.

Zhou, X., Zhuang, X., Yan, S., Chang, S.-F., Hasegawa-Johnson, M., Huang, T. S. (2008). SIFT-Bag Kernel for Video Event Analysis, Proc. 16th ACM Int. Conf. Multimed., 229–238.

A branch and bound algorithm for computing k-nearest neighbours.

Kuffner, A., Robles-Kelly, A. (2006). Image feature evaluation for contents-based image retrieval, Proc. HCSNet Work, 56, 2–6.

Liu, C.-C., Huang, C.-H., Chu, W.-T., Wu, J.-L. (2007). intelligent travel experience management systemITEMS, Proceedings of the international workshop on Workshop on multimedia information retrieval - MIR ’07, 291.

Kennedy, L. S., Naaman, M. (2008). Generating diverse and representative image search results for landmarks, Proceeding 17th Int. Conf, 08-297.

Cootes, T. F., Edwards, G. J., Taylor, C. J. (2001). Active appearance models, IEEE Trans. Pattern Anal. Mach. Intell., 23, 681–685.

Shlens, J. (2014).A Tutorial on Principal Component Analysis.

Okawa, M. (2016). Offline Signature Verification Based on Bag-of-Visual Words Model Using KAZE Features and Weighting Schemes, 184–190.

Yang, G. (1997). Introduction to TCP/IP Network Attacks, Secur. Syst. Lab. Novemb.

Srinoy, S. (2007). Intrusion Detection Model Based On Particle Swarm Optimization and Support Vector Machine, Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Security and Defense Applications, 20.

APPENDICES

Appendix A

Here we outline various Signature recognition algorithm opencv C++ code scripts and functions that were used in this thesis.

SIFT with SVM Code

```
#define _CRT_SECURE_NO_WARNINGS

#include "stdafx.h"

#include <opencv\cv.h>

#include <opencv2\highgui\highgui.hpp>

#include <opencv2\ml\ml.hpp>

#include <stdio.h>

#include <iostream>

#include <opencv2/features2d/features2d.hpp>

#include <opencv2/imgproc/imgproc.hpp>

#include <opencv2/highgui/highgui.hpp>

#include <opencv2/highgui/highgui.hpp>

#include <opencv2/nonfree/features2d.hpp>

#include <vector
```

```
Using namespace c

Using namespace STD;

Using std::cout;

Using std::cerr;

Using std::endl;

Using std::vector;

Char ch [300000

];

Ptr<Descriptor Matcher> matcher = Descriptor Matcher::create ("Flann Based");

Ptr<Descriptor Extractor> extractor = new Sift Descriptor Extractor ();

Sift Feature Detector detector (400);

Int dictionary Size = 1500;

Term Criteria Tc (CV_TERMCRIT_ITER, 10, 0.001);

Int retries = 1;

Int flags = KMEANS_PP_CENTERS;

BOWK Means Trainer bow Trainer (dictionary Size, tc, retries, flags);

BOW Img Descriptor Extractor bow DE (extractor, matcher);

Void collect class centroids () {

IplImage *img;

Int i, j;

For (j = 1; j <= 10; j++)

For (i = 1; i <= 16; i++) {
```

```
Sprintf (ch, "%s%d%s%d%s", "E: /Training Set/Offline Genuine/", j,” (", i,
").png");

Const char* image Name = ch;

Img = cvLoad Image (image Name, 0);

Vector<Keypoint> keypoint;

Detector. Detect (img, keypoint);

Cout << keypoint. Size () << endl;

Mat features;

Extractor->compute (img, keypoint, features);

bowTrainer.add (features);

}

Return;

}

Int _tmain (int argc, _TCHAR* argv [])

{

Int i, j;

IplImage *img2;

Cout << "Vector quantization..." << endl;

Collect class centroids ();

Vector<Mat> descriptors = bowTrainer.getDescriptors ();

Int count = 0;

For (vector<Mat>:: iterator iter = descriptors. Begin (); iter! = descriptors.end ();
iter++)

{
```

```
Count += iter->rows;

}

Cout << "Clustering” << count << “features" << endl;

//choosing cluster's centroids as dictionary's words

Mat dictionary = bowTrainer.cluster ();

bowDE.setVocabulary (dictionary);

Cout << "extracting histograms in the form of BOW for each image” << endl;

Mat labels (0, 1, CV_32FC1);

Mat training Data (0, dictionary Size, CV_32FC1);

Int k = 0;

Vector<Keypoint> keypoint1;

Mat bowDescriptor1;

//extracting histogram in the form of bow for each image

For (j = 1; j <= 10; j++)

For (i = 1; i <= 16; i++) {

Sprintf (ch, "%s%d%s%d%s", "E: /Training Set/Offline Genuine/", j,” (", i,
").png");

Const char* image Name = ch;

img2 = cvLoadImage (image Name, 0);

Cv::Mat input = cv::cvarrToMat (img2);

Detector. Detect (img2, keypoint1);

DrawKeypoints (input, keypoint1, input);

Imshow ("img", input);

Wait Key (1);
```

```
bowDE.compute (img2, keypoint1, bowDescriptor1);

trainingData.push_back (bowDescriptor1);

labels.push_back ((float) j);

}

//Setting up SVM parameters

CvSVMParams params;

params.kernel_type = CvSVM::RBF;

params.svm_type = CvSVM::C_SVC;

params.gamma = 0.50625000000000009;

params.C = 312.50000000000000;

params.term_crit = cvTermCriteria (CV_TERMCRIT_ITER, 100, 0.000001);

CvSVM svm;

Sprintf ("%s\n", "Training SVM classifier");

Bool res = svm.train (training Data, labels, cv::Mat (), cv::Mat (), params);

Cout << "Processing evaluation data..." << endl;

Mat ground Truth (0, 1, CV_32FC1);

Mat evalData (0, dictionary Size, CV_32FC1);

k = 0;

Vector<Keypoint> keypoint2;

Mat bowDescriptor2;

Mat results (0, 1, CV_32FC1);

For (j = 1; j <= 10; j++)

For (i = 17; i <= 24; i++) {
```

```
Sprintf (ch, "%s%d%s%d%s", "E: /Training Set/Offline Genuine//", j,” (", i,
").png");

Const char* image Name = ch;

img2 = cvLoadImage (image Name, 0);

Detector. Detect (img2, keypoint2);

bowDE.compute (img2, keypoint2, bowDescriptor2);

evalData.push_back (bowDescriptor2);

groundTruth.push_back ((float)j);

Float response = svm. Predict (bowDescriptor2);

results.push_back (response);

Cout << response << "   ==“+ to_string (j) + " (" + to_string (i) + ").png" <<
endl;

}

//calculate the number of unmatched classes

Double error Rate = (double) count Nonzero (ground Truth - results) / evalData.
Rows;

Cout << (double) count Nonzero (ground Truth - results) << endl;

Sprintf ("%s%f", "Error rate is ", error Rate);

System ("pause");

return 0;          //

//

Return 0;
```

SURF with SVM Code

```
#define _CRT_SECURE_NO_WARNINGS

#include "stdafx.h"

#include <opencv\cv.h>

#include <opencv2\highgui\highgui.hpp>

#include <opencv2\ml\ml.hpp>

#include <stdio.h>

#include <iostream>

#include <opencv2/features2d/features2d.hpp>

#include <opencv2/imgproc/imgproc.hpp>

#include <opencv2/highgui/highgui.hpp>

#include <opencv2/highgui/highgui.hpp>

#include <opencv2/nonfree/features2d.hpp>

#include <vector>

Using namespace CV;

Using namespace STD;

Using std::cout;

Using std::cerr;

Using std::endl;

Using std::vector;

Char ch [300000000];

Ptr<Descriptor Matcher> matcher = Descriptor Matcher::create ("Flann Based");

Ptr<Descriptor Extractor> extractor = new Surf Descriptor Extractor ();
```

```
Surf Feature Detector detector (500);

Int dictionary Size = 1500;

Term Criteria Tc (CV_TERMCRIT_ITER, 10, 0.001);

Int retries = 1;

Int flags = KMEANS_PP_CENTERS;

BOWKMeansTrainer bow Trainer (dictionary Size, tc, retries, flags);

BOWImgDescriptorExtractor bow DE (extractor, matcher);

Void collectclasscentroids () {

IplImage *img;

Int i, j;

For (j = 1; j <= 10; j++)

For (i = 1; i <= 16; i++) {

Sprintf (ch, "%s%d%s%d%s", "E: /Training Set/Offline Genuine/", j,”
(", i, ").png");

Const char* image Name = ch;

Img = cvLoadImage (image Name, 0);

Vector<Keypoint> keypoint;

Detector. Detect (img, keypoint);

Cout << keypoint. Size () << endl;

Mat features;

Extractor->compute (img, keypoint, features);

bowTrainer.add (features);

}
```

```
Return;

}

Int _tmain (int argc, _TCHAR* argv [])

{

Int i, j;

IplImage *img2;

Cout << "Vector quantization..." << endl;

Collectclasscentroids ();

Vector<Mat> descriptors = bowTrainer.getDescriptors ();

Int count = 0;

For (vector<Mat>:: iterator iter = descriptors. Begin (); iter! = descriptors.end
(); iter++)

{

Count += iter->rows;

}

Cout << "Clustering” << count << “features" << endl;

//choosing cluster's centroids as dictionary's words

Mat dictionary = bowTrainer.cluster ();

bowDE.setVocabulary (dictionary);

Cout << "extracting histograms in the form of BOW for each image” << endl;

Mat labels (0, 1, CV_32FC1);

Mat training Data (0, dictionary Size, CV_32FC1);

Int k = 0;
```

```
Vector<Keypoint> keypoint1;

Mat bowDescriptor1;

//extracting histogram in the form of bow for each image

For (j = 1; j <= 10; j++)

For (i = 1; i <= 16; i++) {

Sprintf (ch, "%s%d%s%d%s", "E: /TrainingSet/Offline Genuine/", j,”
(", i, ").png");

Const char* image Name = ch;

img2 = cvLoadImage (image Name, 0);

Cv::Mat input = cv::cvarrToMat (img2);

Detector. Detect (img2, keypoint1);

DrawKeypoints (input, keypoint1, input);

Imshow ("img", input);

Wait Key (1);

bowDE.compute (img2, keypoint1, bowDescriptor1);

trainingData.push_back (bowDescriptor1);

labels.push_back ((float) j);

}

//Setting up SVM parameters

CvSVMParams params;

params.kernel_type = CvSVM::RBF;

params.svm_type = CvSVM::C_SVC;

params.gamma = 0.50625000000000009;
params.C = 312.50000000000000;

params.term_crit = cvTermCriteria (CV_TERMCRIT_ITER, 100, 0.000001);

CvSVM svm;

Sprintf ("%s\n", "Training SVM classifier");

Bool res = svm.train (training Data, labels, cv::Mat (), cv::Mat (), params);

Cout << "Processing evaluation data..." << endl;

Mat ground Truth (0, 1, CV_32FC1);

Mat evalData (0, dictionary Size, CV_32FC1);

k = 0;

Vector<Keypoint> keypoint2;

Mat bowDescriptor2;

Mat results (0, 1, CV_32FC1);

For (j = 1; j <= 10; j++)

For (i = 17; i <= 24; i++) {

Sprintf (ch, "%s%d%s%d%s", "E: /TrainingSet/Offline Genuine//", j,”
(", i, ").png");

Const char* image Name = ch;

img2 = cvLoadImage (image Name, 0);

Detector. Detect (img2, keypoint2);

bowDE.compute (img2, keypoint2, bowDescriptor2);

evalData.push_back (bowDescriptor2);

groundTruth.push_back ((float) j);

Float response = svm. Predict (bowDescriptor2);
results.push_back (response);

Cout << response << "   ==“+ to_string (j) + " (" + to_string (i) +
").png" << endl;

}

//calculate the number of unmatched classes

Double error Rate = (double) count Nonzero (ground Truth - results) /
evalData. Rows;

Cout << (double) count Nonzero (ground Truth - results) << endl;

Sprintf ("%s%f", "Error rate is ", error Rate);

System ("pause");

return 0;         //

//

Return
```

Client for Voting Online

```
Import java.awt.Color;

Import java.awt.Dimension;

Import java.awt.Font;

Import java.awt.Image;

Import java.awt.Toolkit;

Import java.awt.event.ActionEvent;

Import java.awt.event.ActionListener;

Import java.awt.image.BufferedImage;

Import java.io.BufferedReader;

Import java.io.BufferedWriter;

Import java.io.ByteArrayOutputStream;

Import java.io.DataInputStream;

Import java.io.DataOutputStream;

Import java.io.File;

Import java.io.FileInputStream;

Import java.io.FileNotFoundException;

Import java.io.FileReader;

Import static java.nio.file.StandardWatchEventKinds.ENTRY_CREATE;

Import static java.nio.file.StandardWatchEventKinds.ENTRY_DELETE;

Import static java.nio.file.StandardWatchEventKinds.ENTRY_MODIFY;
Import java.io.IOException;

Import java.io.InputStream;

Import java.io.InputStreamReader;

Import java.io.OutputStream;

Import java.io.OutputStreamWriter;

Import java.net.InetAddress;

Import java.net.ServerSocket;

Import java.net.Socket;

Import java.net.SocketTimeoutException;

Import java.nio.file.FileSystems;

Import java.nio.file.Path;

Import java.nio.file.Paths;

Import java.nio.file.WatchEvent;

Import java.nio.file.WatchKey;

Import java.nio.file.WatchService;

Import java.sql.Connection;

Import java.sql.PreparedStatement;

Import java.sql.ResultSet;

Import java.sql.SQLException;

Import java.sql.Statement;

Import java.text.DateFormat;

Import java.text.ParseException;

Import java.text.SimpleDateFormat;
Import java.util.Date;

Import javax.imageio.ImageIO;

Import javax.swing.ImageIcon;

Import javax.swing.JButton;

Import javax.swing.JComboBox;

Import javax.swing.JFileChooser;

Import javax.swing.JFrame;

Import javax.swing.JLabel;

Import javax.swing.JLabel;

Import javax.swing.JOptionPane;

Import javax.swing.border.MatteBorder;

Import oracle.jdbc.pool.OracleDataSource;

Import sun. Audio;

Public class Client

Public static String filename;

Public static Socket s;

Public static Socket socket;

Public static String linebuff;

Static public JFrame ServerFrm;

Public static Matte Border Matte;

Public   static JButton TrainingDBINSERTbtn;

public static JLabel
IDimg,ID,Name,Father_name,Mother_name,Address,Birth_Date,Age,Job_addr,Bloo
d_type,ok;
```

```
// public static JLabel State1;

Public static void run () throws SQLException

{

Thread t= new Thread ();

ServerFrm=new JFrame ("Client Window"); //create frame and named

ServerFrm.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);//when end,
frame exit from program

ServerFrm.setLayout (null); //remove layout

ServerFrm.setSize (1000, 1000); //set size of frame [762, 456]

ServerFrm.setResizable (false); //not resizable

Matte=new Matte Border (2, 2, 2, 2, Color. Black); //set border type (matte border)

TrainingDBINSERTbtn=new JButton (); //create button for Start camera capture

TrainingDBINSERTbtn.setText ("insert"); //set label name in button

TrainingDBINSERTbtn.setBounds (600, 600, 100, 30); //set bounds in frame

TrainingDBINSERTbtn.addActionListener (new Client.InsertDB ()); //

IDimg=new JLabel (); //create label for ID

IDimg.setOpaque (true);

IDimg.setBackground (Color. White); //set color to jlabel

IDimg.setBorder (Matte); //set border in jlabel

IDimg.setBounds (400,100,512,350); //set bounds in frame

ID=new JLabel (); //create label for ID

ID.setOpaque (true);

ID.setBackground (Color. White);//set color to jlabel
ID.setBorder (Matte); //set border in jlabel

ID.setBounds (105, 105, 100, 20);

Name=new JLabel ();

Name.setOpaque (true);

Name.setBackground (Color. White);

Name.setBorder (Matte);

Name.setBounds (105, 155, 100, 20);

Father name=new JLabel ();

Father_name.setOpaque (true);


Father_name.setBackground (Color.white);

Father_name.setBorder (Matte);

Father_name.setBounds (105, 205, 150, 20);

Mother name=new JLabel ();

Mother_name.setOpaque (true);

Mother_name.setBackground (Color.white);

Mother_name.setBorder (Matte);

Mother_name.setBounds (105, 255, 150, 20);

Birthdate=new JLabel ();

Birth_Date.setOpaque (true);

Birth_Date.setBackground (Color.white);

Birth_Date.setBorder (Matte);

Birth_Date.setBounds (105, 305, 150, 20); //set bounds in frame
```

```
Age=new JLabel (); //create label for ID

Name.setOpaque (true);

Age.setBackground (Color.white);//set color to jlabel

Name.setBorder (Matte); //set border in jlabel

Age.setBounds (105, 355, 150, 20);//set bounds in frame

Job_addr=new JLabel (); //create label for ID

Job_addr.setOpaque (true);

Job_addr.setBackground (Color.white);//set color to jlabel

Job_addr.setBorder (Matte); //set border in jlabel

Job_addr.setBounds (105, 405, 150, 20); //set bounds in frame

Blood type=new JLabel (); //create label for Name

Blood_type.setOpaque (true);

Blood_type.setBackground (Color.white);//set color to jlabel

Blood_type.setBorder (Matte); //set border in jlabel

Blood_type.setBounds (105, 455, 150, 20); //set bounds in frame

Address=new JLabel (); //create label for Name

Address.setOpaque (true);

Address.setBackground (Color.white);//set color to jlabel

Address.setBorder (Matte); //set border in jlabel

Address.setBounds (105, 505, 150, 20); //set bounds in frame

JLabel TrainingNameTitlelbl=new JLabel ("ID :"); //create title for Name

TrainingNameTitlelbl.setOpaque (true);

TrainingNameTitlelbl.setFont (new Font ("Tahoma", Font.BOLD,13));//set font
TrainingNameTitlelbl.setBounds (5, 100, 60, 30); //set bounds in frame

JLabel agelbl=new JLabel ("Name :"); //create title for Name

agelbl.setOpaque (true);

agelbl.setFont (new Font ("Tahoma", Font.BOLD,13));//set font

agelbl.setBounds (5, 150, 60, 30); //set bounds in frame

JLabel birthlbl=new JLabel ("Father name :"); //create title for Name

birthlbl.setOpaque (true);

birthlbl.setFont (new Font ("Tahoma", Font.BOLD,13));//set font

birthlbl.setBounds (5, 200, 100, 30); //set bounds in frame

JLabel statelbl=new JLabel ("Mother name :"); //create title for Name

statelbl.setOpaque (true);

statelbl.setFont (new Font ("Tahoma", Font.BOLD,13));//set font

statelbl.setBounds (5, 250, 100, 30); //set bounds in frame

JLabel jobtitlbl=new JLabel ("Birth date :"); //create title for Name

jobtitlbl.setOpaque (true);

jobtitlbl.setFont (new Font ("Tahoma", Font.BOLD,13));//set font

jobtitlbl.setBounds (5, 300, 100, 30); //set bounds in frame

JLabel jobadrrlbl=new JLabel ("Age :"); //create title for Name

jobadrrlbl.setOpaque (true);

jobadrrlbl.setFont (new Font ("Tahoma", Font. BOLD, 13));

jobadrrlbl.setBounds (5, 350, 100, 30);

JLabel Maillol=new JLabel ("Job :");

Emaillbl.setOpaque (true);
Emaillbl.setFont (new Font ("Tahoma", Font. BOLD, 13));

Emaillbl.setBounds (5, 400, 100, 30);

JLabel addrlbl=new JLabel ("Blood Type :");

addrlbl.setOpaque (true);

addrlbl.setFont (new Font ("Tahoma", Font.BOLD, 13));

addrlbl.setBounds (5, 450, 100, 30);

JLabel addrlbl=new JLabel ("Address :");

addrwlbl.setOpaque (true);

addrlbl.setFont (new Font ("Tahoma", Font. BOLD, 13));

addrlbl.setBounds (5, 500, 100, 30);

ServerFrm. Add (TrainingDBINSERTbtn);

ServerFrm. Add (IDimg);

ServerFrm. Add (ID);

ServerFrm. Add (Birthdate);

ServerFrm. Add (Mother_name);

ServerFrm. Add (Father_name);

ServerFrm. Add (Job_addr);

ServerFrm. Add (Name);

ServerFrm. Add (Age);

ServerFrm. Add (Blood_type);

ServerFrm. Add (Address);

ServerFrm. Add (TrainingNameTitlelbl);

ServerFrm. Add (agelbl);
ServerFrm. Add (birthlbl);

ServerFrm. Add (statelbl);

ServerFrm. Add (addrlbl);

ServerFrm. Add (jobadrrlbl);

ServerFrm. Add (jobtitlbl);

ServerFrm. Add (Emaillbl);

ServerFrm. Add (addrwlbl);

ServerFrm.setVisible (true);

}

Public static void main (String [] args) throws ParseException {

Try {

Run ();

} catch (Exception e) {

System.err.println ("error server");

}

Try

{

Int port = 25000;

Server Socket server Socket = new Server Socket (port);

System.out.println ("Server Started and listening to the port 25000");

While (true)

{

Socket = serverSocket.accept ();
Input Stream is = socket.getInputStream ();

InputStreamReader isr = new InputStreamReader (is);

Buffered Reader br = new BufferedReader (isr);

String data = br.readLine ();

System.out.println ("Message received from client is "+data);

ID.setText (data.substring (data.indexOf ("f1") +2, data.indexOf ("f2")));

Name.setText (data.substring (data.indexOf ("f2") +2, data.indexOf ("f3")));

Father_name.setText (data.substring (data.indexOf ("f3") +2, data.indexOf ("f4")));

Mother_name.setText (data.substring (data.indexOf ("f4") +2, data.indexOf ("f5")));

Birth_Date.setText (data.substring (data.indexOf ("f5") +2, data.indexOf ("f6")));

Age.setText (data.substring (data.indexOf ("f6") +2, data.indexOf ("f7")));

Job_addr.setText (data.substring (data.indexOf ("f7") +2, data.indexOf ("f8")));

Blood_type.setText (data.substring (data.indexOf ("f8") +2, data.indexOf ("f9")));

Address.setText (data.substring (data.indexOf ("f9") +2));

}

}

Catch (Exception e)

{

e.printStackTrace ();

}

Finally

{

Try
```

```
{
socket.close ();
}
Catch (Exception e) {}
}
}
Public static class InsertDB implements ActionListener {
Public void actionPerformed (Action Event e) {
Buffered Image img = null;
Try {
JFileChooser chooser=new JFileChooser ();
chooser.showOpenDialog (null);
chooser.getSelectedFile ();
File f=chooser.getSelectedFile ();
Filename=f.getAbsolutePath ();
System.out.println (filename);
Img = ImageIO.read (new File (filename));
} catch (Exception ee) {
JOptionPane.showMessageDialog (null, "Please Choose Image", "Error”, JOptionPane.ERROR_MESSAGE);

System.out.println ("Please Choose Image");
}
```

```
ImageIcon icon = new ImageIcon (filename);

Image image = icon.getImage ();

Image newimg = image.getScaledInstance (IDimg.getWidth (), IDimg.getHeight (),
java.awt.Image.SCALE_SMOOTH); // scale it the smooth way

ImageIcon icon1 = new ImageIcon (newimg);

IDimg.setIcon (icon1);

Try {

s=new Socket ("127.0.0.1", 6066);

Output Stream os=s.getOutputStream ();

Img=ImageIO.read (new File (filename));

ByteArrayOutputStream baos=new ByteArrayOutputStream ();

ImageIO.write (img, "png", baos);

os.write (baos.toByteArray ());

Os. Flush ();

JOptionPane.showMessageDialog (null, "Sent");

} catch (IOException eee) {

System.err.println ("error socket");

}


}


}
         {
```

Server of Voting Online Code

```
Import java.awt.Color;
Import java.awt.Dimension;
Import java.awt.Font;
Import java.awt.Image;
Import java.awt.Toolkit;
Import java.awt.image.BufferedImage;
Import java.io.BufferedReader;
Import java.io.BufferedWriter;
Import java.io.DataInputStream;
Import java.io.DataOutputStream;
Import java.io.File;
Import java.io.FileInputStream;
Import java.io.FileNotFoundException;
Import java.io.FileReader;
Import static java.nio.file.StandardWatchEventKinds.ENTRY_CREATE;
Import static java.nio.file.StandardWatchEventKinds.ENTRY_DELETE;
Import static java.nio.file.StandardWatchEventKinds.ENTRY_MODIFY;
Import java.io.IOException;
Import java.io.InputStream;
Import java.io.InputStreamReader;
Import java.io.OutputStream;
Import java.io.OutputStreamWriter;
Import java.net.InetAddress;
Import java.net.ServerSocket;
Import java.net.Socket;
Import java.net.SocketTimeoutException;
Import java.nio.file.FileSystems;
Import java.nio.file.Path;
Import java.nio.file.Paths;
Import java.nio.file.WatchEvent;
Import java.nio.file.WatchKey;
Import java.nio.file.WatchService;
Import java.sql.Connection;
Import java.sql.ResultSet;
Import java.sql.SQLException;
Import java.sql.Statement;
Import java.text.DateFormat;
Import java.text.ParseException;
Import java.text.SimpleDateFormat;
Import java.util.Date;
Import javax.imageio.ImageIO;
Import javax.swing.ImageIcon;
Import javax.swing.JButton;
Import javax.swing.JComboBox;
Import javax.swing.JFrame;
Import javax.swing.JLabel;
Import javax.swing.JLabel;
Import javax.swing.JOptionPane;
Import javax.swing.border.MatteBorder;
Import oracle.jdbc.pool.OracleDataSource;
//import oracle.jdbc.pool.OracleDataSource;
 Import sun. Audio.*;

Public class Server {
Public static String linebuff;
Static public JFrame ServerFrm;
 Public static MatteBorder Matte;
```

```
public static JLabel
IDimg,ID,Name,Father_name,Mother_name,Address,Birth_Date,Age,Job_addr,Bloo
d_type,ok;

 // public static JLabel State1;

   Private static ServerSocket serverSocket;

   Static Socket server;

  Public static void Greeting Server (int port) throws IOException, SQLException,
ClassNotFoundException, and Exception

      {

         ServerSocket = new ServerSocket (port);

         serverSocket.setSoTimeout (18000000);


      }

      Public static Socket socket, socket1;


      Public static void run () throws SQLException

      {

                      Thread t= new Thread ();

                        ServerFrm=new JFrame ("Server Window");//create frame and
named


        ServerFrm.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);//when end,
frame exit from program


  ServerFrm.setLayout (null); //remove layout

        ServerFrm.setSize (1000, 1000); //set size of frame [762, 456]


        ServerFrm.setResizable (false); //not resizable
        Matte=new MatteBorder (2, 2, 2, 2, Color.black);//set border type (matte
    border)
        IDimg=new JLabel (); //create label for ID
        IDimg.setOpaque (true);
        IDimg.setBackground (Color.white);//set color to jlabel
        IDimg.setBorder (Matte); //set border in jlabel
        IDimg.setBounds (400,100,512,350); //set bounds in frame
        ID=new JLabel (); //create label for ID
        ID.setOpaque (true);
        ID.setBackground (Color.white);//set color to jlabel
        ID.setBorder (Matte); //set border in jlabel
        ID.setBounds (105, 105, 100, 20); //set bounds in frame

       Name=new JLabel (); //create label for Name
        Name.setOpaque (true);
        Name.setBackground (Color.white);//set color to jlabel
        Name.setBorder (Matte); //set border in jlabel
        Name.setBounds (105, 155, 100, 20); //set bounds in frame
     //
        Father_name=new JLabel (); //create label for Name
        Father_name.setOpaque (true);
        Father_name.setBackground (Color.white);//set color to jlabel
        Father_name.setBorder (Matte); //set border in jlabel
        Father_name.setBounds (105, 205, 150, 20); //set bounds in frame
    //
        Mother_name=new JLabel (); //create label for ID
        Mother_name.setOpaque (true);
        Mother_name.setBackground (Color.white);//set color to jlabel
        Mother_name.setBorder (Matte); //set border in jlabel
        Mother_name.setBounds (105, 255, 150, 20); //set bounds in frame
        Birth_Date=new JLabel (); //create label for ID
        Birth_Date.setOpaque (true);
        Birth_Date.setBackground (Color.white);//set color to jlabel
        Birth_Date.setBorder (Matte); //set border in jlabel
        Birth_Date.setBounds (105, 305, 150, 20);//set bounds in frame
//      ////////////////////
//      /////////////
//
        Age=new JLabel (); //create label for ID
        Name.setOpaque (true);
        Age.setBackground (Color.white);//set color to jlabel
        Name.setBorder (Matte); //set border in jlabel
        Age.setBounds (105, 355, 150, 20); //set bounds in frame

        /////////////
        Job_addr=new JLabel (); //create label for ID
        Job_addr.setOpaque (true);
        Job_addr.setBackground (Color.white);//set color to jlabel
        Job_addr.setBorder (Matte); //set border in jlabel
        Job_addr.setBounds (105, 405, 150, 20); //set bounds in frame
        ////////////////////
//
//
//
        Blood_type=new JLabel (); //create label for Name
        Blood_type.setOpaque (true);
        Blood_type.setBackground (Color.white);//set color to jlabel
        Blood_type.setBorder (Matte); //set border in jlabel
        Blood_type.setBounds (105, 455, 150, 20); //set bounds in frame

//     //
         Address=new JLabel (); //create label for Name
        Address.setOpaque (true);
        Address.setBackground (Color.white);//set color to jlabel
        Address.setBorder (Matte); //set border in jlabel
        Address.setBounds (105, 505, 150, 20); //set bounds in frame

        JLabel TrainingNameTitlelbl=new JLabel ("ID :"); //create title for Name
        TrainingNameTitlelbl.setOpaque (true);
        TrainingNameTitlelbl.setFont (new Font ("Tahoma", Font.BOLD,13));//set font
        TrainingNameTitlelbl.setBounds (5, 100, 60, 30); //set bounds in frame

//
//
        JLabel agelbl=new JLabel ("Name :"); //create title for Name
        agelbl.setOpaque (true);
        agelbl.setFont (new Font ("Tahoma", Font.BOLD,13));//set font
        agelbl.setBounds (5, 150, 60, 30); //set bounds in frame
//
        JLabel birthlbl=new JLabel ("Father name :"); //create title for Name
        birthlbl.setOpaque (true);
        birthlbl.setFont (new Font ("Tahoma", Font.BOLD,13));//set font
        birthlbl.setBounds (5, 200, 100, 30); //set bounds in frame



        JLabel statelbl=new JLabel ("Mother name :"); //create title for Name
```

```
         statelbl.setOpaque (true);
         statelbl.setFont (new Font ("Tahoma", Font.BOLD,13));//set font
         statelbl.setBounds (5, 250, 100, 30); //set bounds in frame
       JLabel jobtitlbl=new JLabel ("Birth date :"); //create title for Name
         jobtitlbl.setOpaque (true);
         jobtitlbl.setFont (new Font ("Tahoma", Font.BOLD,13));//set font
         jobtitlbl.setBounds (5, 300, 100, 30); //set bounds in frame
//
          JLabel jobadrrlbl=new JLabel ("Age :"); //create title for Name
         jobadrrlbl.setOpaque (true);
         jobadrrlbl.setFont (new Font ("Tahoma", Font.BOLD,13));//set font
         jobadrrlbl.setBounds (5, 350, 100, 30); //set bounds in frame
//
          JLabel Emaillbl=new JLabel ("Job :"); //create title for Name
         Emaillbl.setOpaque (true);
         Emaillbl.setFont (new Font ("Tahoma", Font.BOLD,13));//set font
         Emaillbl.setBounds (5, 400, 100, 30); //set bounds in frame
         JLabel addrlbl=new JLabel ("Blood Type :"); //create title for Name
         addrlbl.setOpaque (true);
         addrlbl.setFont (new Font ("Tahoma", Font.BOLD,13));//set font
         addrlbl.setBounds (5, 450, 100, 30); //set bounds in frame

         JLabel addrwlbl=new JLabel ("Address :"); //create title for Name
         addrwlbl.setOpaque (true);
         addrlbl.setFont (new Font ("Tahoma", Font.BOLD,13));//set font
         addrlbl.setBounds (5, 500, 100, 30); //set bounds in frame

         //Birth_Date,State,Job_tit,Job_addr
        ServerFrm.add (IDimg); //add IDlbl to frame
        ServerFrm.add (ID); //add IDlbl to frame
         ServerFrm.add (Birth_Date); //add IDlbl to frame
         ServerFrm.add (Mother_name); //add IDlbl to frame
         ServerFrm.add (Father_name); //add IDlbl to frame
         ServerFrm.add (Job_addr); //add IDlbl to frame
         ServerFrm.add (Name); //add Namelbl to frame
        ServerFrm.add (Age); //add IDlbl to frame
        ServerFrm.add (Blood_type); //add IDlbl to frame
         ServerFrm.add (Address); //add IDlbl to frame
         ServerFrm.add (TrainingNameTitlelbl);//add Name title to frame
         ServerFrm.add (agelbl); //add Nametitle to frame
         ServerFrm.add (birthlbl);
         ServerFrm.add (statelbl);
         ServerFrm.add (addrlbl);
         ServerFrm.add (jobadrrlbl);
         ServerFrm.add (jobtitlbl);
         ServerFrm.add (Emaillbl);
         ServerFrm.add (addrwlbl);

       // frm.add (ExitTimeTitlelbl); //add ExitTimeTitlelbl to frame

       //========================image label
chooser===============================

         ServerFrm.setVisible (true); //show it
           While (true)
          {
              Try
              {
                Server = serverSocket.accept ();
```

```
              DataInputStream din=new DataInputStream (server.getInputStream ());
              DataOutputStream dout=new DataOutputStream
(server.getOutputStream ());


              Buffered Image img=ImageIO.read (ImageIO.createImageInputStream
(server.getInputStream ()));
                   ImageIO.write (img,"png", new File ("dd.png"));
                                  System.err.println (img.getWidth () +     "
"+img.getHeight ());



//              ImageIcon icon = new ImageIcon ("dd.png");
//Image image = icon.getImage (); // transform it
Image newimg = image.getScaledInstance (IDimg.getWidth (), IDimg.getHeight (),
java.awt.Image.SCALE_SMOOTH); // scale it the smooth way
ImageIcon icon1 = new ImageIcon (newimg);


IDimg.setIcon (icon1);
   //lblimg.setIcon (img);


      If (img.getWidth ()>0&&img.getHeight ()>0) {
                              System.out.println ("Image received!!!!");


              }
               JOptionPane.showMessageDialog (null, "Image received!!!!");
                String filePath =
"E:\\abdalbarry\\eval\\x64\\Release\\Face_Rec_Sys.exe";


If (new File (filePath).exists ()) {
   Try {
      ProcessBuilder Pb = new ProcessBuilder (filePath);
```

```
        pb.redirectError ();
        Process p = pb.start ();
        InputStream is = p.getInputStream ();
        Int value = -1;
        While ((value = is.read ())! = -1) {
           System.err.println ((char) value);
        }

        Int exitCode = p.waitFor ();

       // System.out.println (filePath + “exited with” + exitCode);

     } catch (Exception e) {
        e.printStackTrace ();
     }
  } else {
     System.err.println (filePath + “does not exist");
  } // TODO add your handling code here
  /////////////////////////////////////////////////////////////////////////////
        String filename = "res.txt";

        String line = null;

        Try {
           // File Reader reads text files in the default encoding.
           FileReader file Reader =
              New FileReader (fileName);
           // always wrap FileReader in BufferedReader.
           BufferedReader BufferedReader =
            New BufferedReader (file Reader);


          While ((line = bufferedReader.readLine ())! = null) {
            System.out.println (line);
            Linebuff=line;
          }


          // always close files.
          bufferedReader.close ();
        }
        Catch (FileNotFoundException exx) {
          System.out.println (
            "Unable to open file '" +
            fileName + "'");
        }
        Catch (IOException exx) {
          System.out.println (
            "Error reading file '"
            + fileName + "'");
          // or we could just do this:
          // ex.printStackTrace ();
        }


                    OracleDataSource ods = new OracleDataSource ();
ods.setURL ("jdbc:oracle:thin:scott/tiger@localhost:1521:orcl");
Connection conn = ods.getConnection ();
Statement stmt = conn.createStatement ();
             System.err.println (linebuff);
            // String fileName;
```

```
 ResultSet rset = stmt.executeQuery ("SELECT ID_Num, pName, fath_name,
mother_name, Age, address, Blood_type, occupation, birth_date FROM sig where
ID_Num="+linebuff);


 Statement st=conn.createStatement ();
 Int cc=1;
While (rset. Next ()) {
    String userid=rset.getString ("ID_Num");
    Int userint=Integer.valueOf (userid);
    ID.setText (String.valueOf (userint));
 //   System.err.println (userid);


    String name=rset.getString ("pName");
    Name.setText (name);
   //     System.err.println (name);
     String age=rset.getString ("fath_name");
   // int ageint=Integer.valueOf (age);
    Father_name.setText (age);
     String BIRTH=rset.getString ("mother_name");
    Mother_name.setText (BIRTH);
     String STATE=rset.getString ("Age");
    Age.setText (String.valueOf (STATE));


     String JOB=rset.getString ("address");
    Address.setText (String.valueOf (JOB));


     String JOB_ADDRESS=rset.getString ("Blood_type");
    Blood_type.setText (String.valueOf (JOB_ADDRESS));


 String email=rset.getString ("occupation");
```

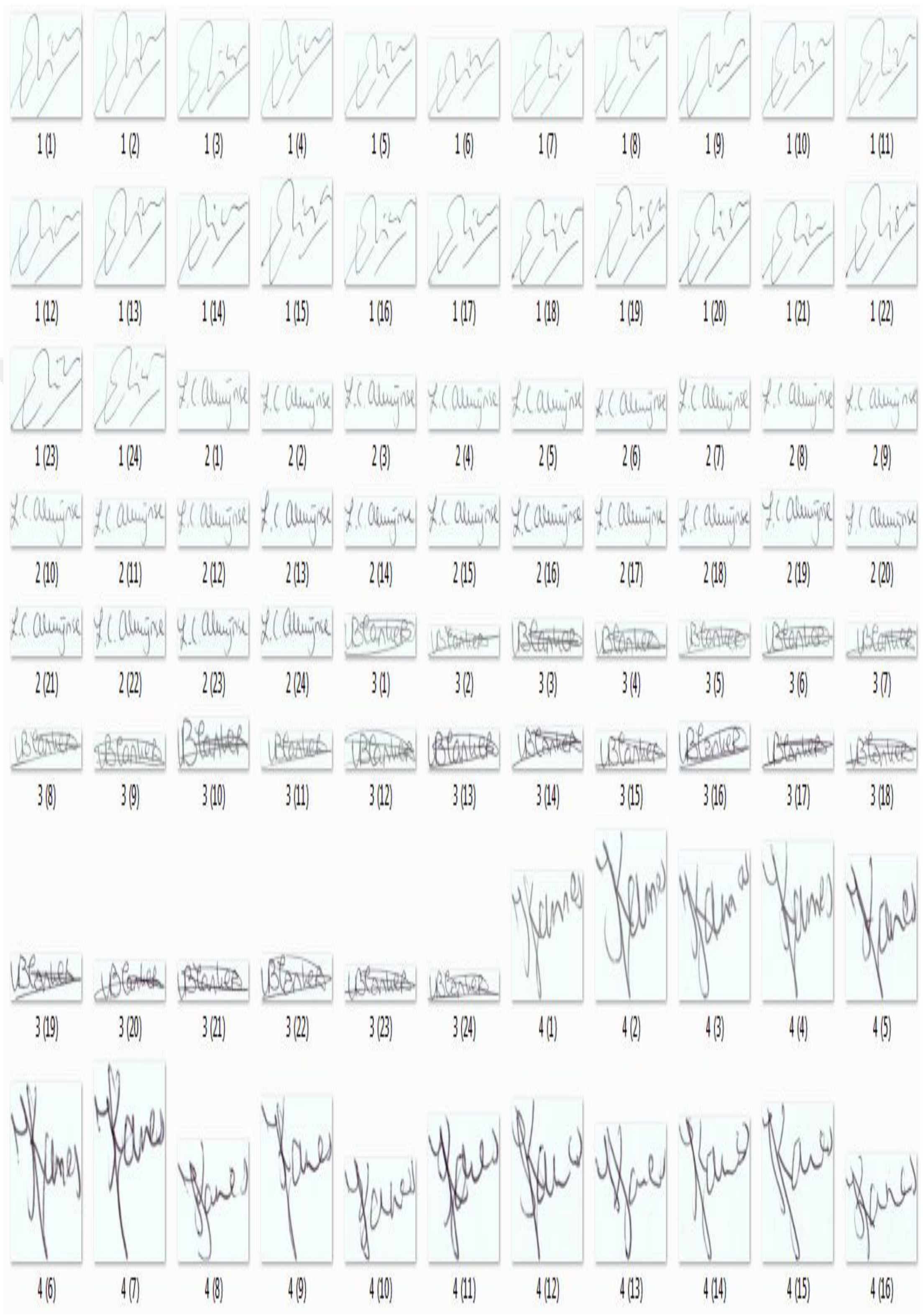
```
    Job_addr.setText (email);


     String address=rset.getString ("birth_date");
    Birth_Date.setText (address);


//     jLabel2.repaint ();
//              jLabel2.revalidate ();
//


}


rset.close ();
st.close ();
 stmt.close ();
 conn.close ();
Try
        {
            String host = "127.0.0.1";
            Int port = 25000;
            InetAddress address = InetAddress.getByName (host);
            Socket = new Socket (address, port);


            //Send the message to the server
            OutputStream os = socket.getOutputStream ();
            OutputStreamWriter osw = new OutputStreamWriter (os);
            BufferedWriter BW = new BufferedWriter (osw);
            String sendMessage = 
"f1"+ID.getText()+"f2"+Name.getText()+"f3"+Father_name.getText()+"f4"+Mother_
name.getText()+"f5"+Birth_Date.getText()+"f6"+Age.getText()+"f7"+Job_addr.getTe
xt()+"f8"+Blood_type.getText()+"f9"+Address.getText() ;
```

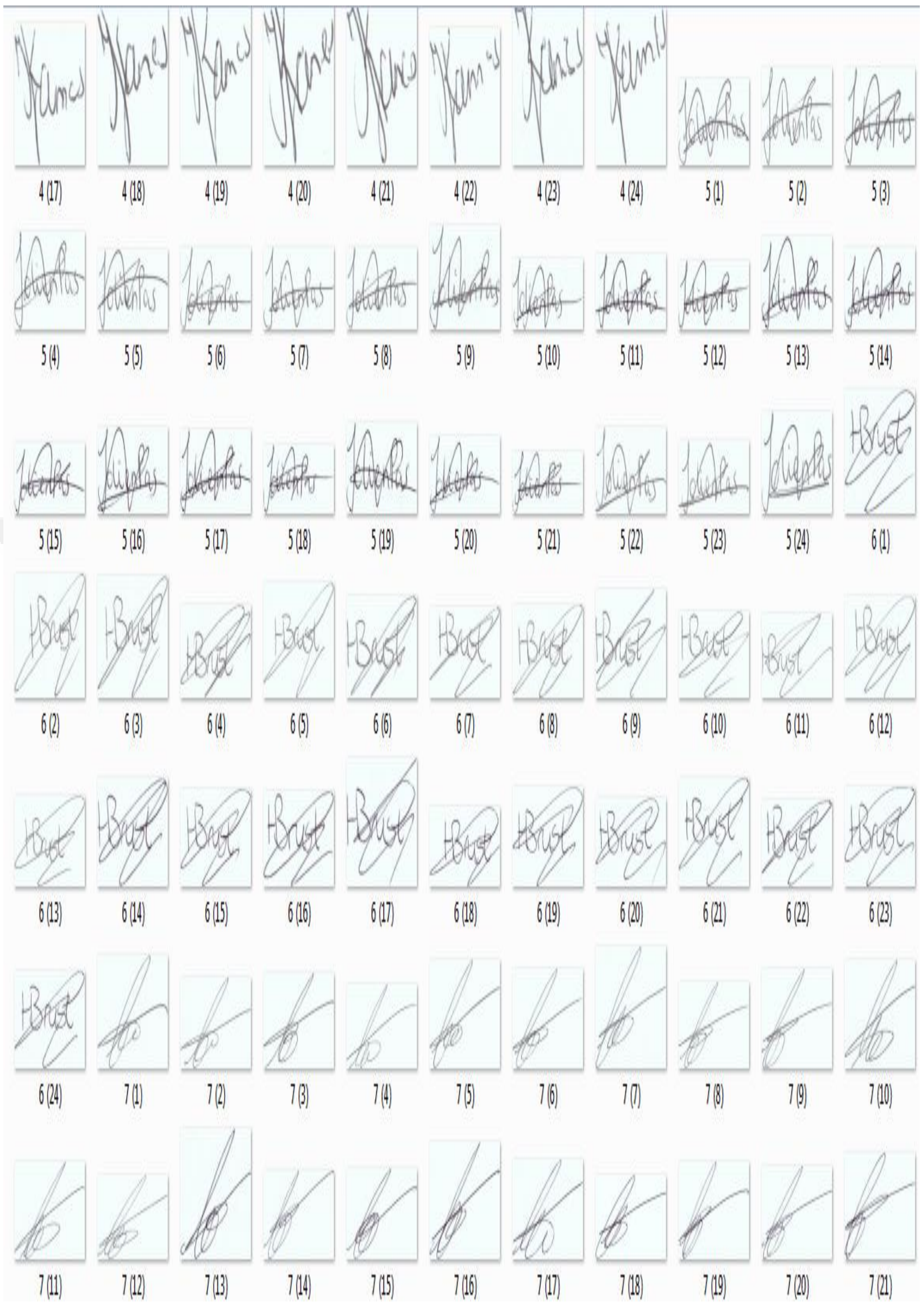
```
   BW. Write (sendMessage);
   BW. Flush ();
   System.out.println ("Message sent to the server: "+sendMessage);
}
Catch (Exception exception)
{
   exception.printStackTrace ();
}
Finally
{
   //Closing the socket
   Try
   {
      socket.close ();
   }
   Catch (Exception e)
   {
      e.printStackTrace ();
   }
}
    }
    Catch (SocketTimeoutException st)
    {
        System.out.println ("Socket timed out!");
        Break;
    }
    Catch (IOException e)
    {
                    System.out.println (e);
```

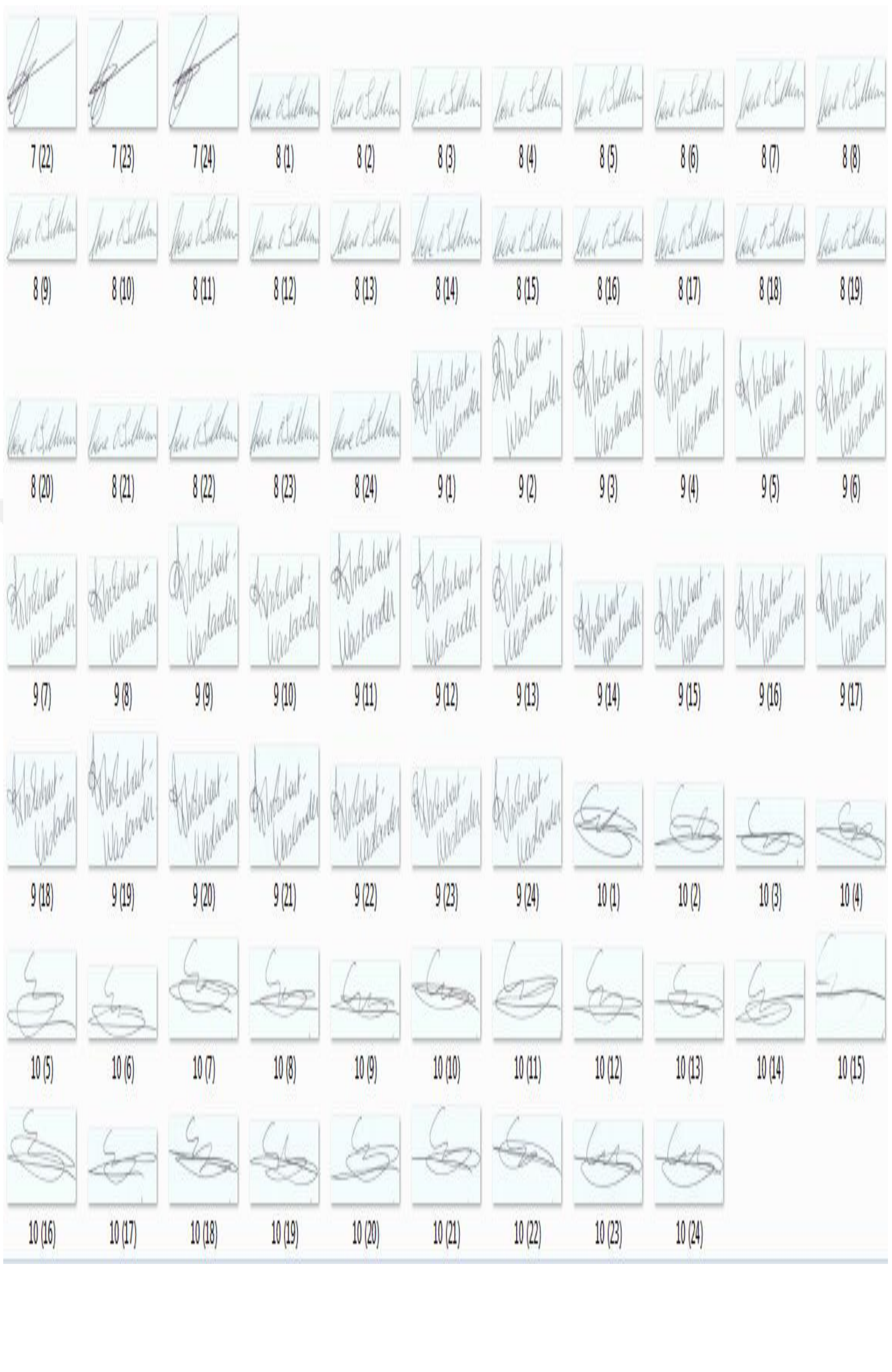
```
                e.printStackTrace ();
                Break;
            }
            Catch (Exception ex)
           {
                System.out.println (ex);
           }
         }
       }

    Public static void main (String [] args) throws ParseException {
       Try {
          GreetingServer (6066); ///////////////////////////////////////////
        // Thread t = new GreetingServer (6066);
          // t.start ();
            Run ()
       } catch (Exception e) {
                System.err.println ("error server");
       }
    }
}
```

Appendix B

Shows Signatures Used in this Thesis




4 (17)
4 (18)
4 (19)
4 (20)
4 (21)
4 (22)
4 (23)
4 (24)
5 (1)
5 (2)
5 (3)
5 (4)
5 (5)
5 (6)
5 (7)
5 (8)
5 (9)
5 (10)
5 (11)
5 (12)
5 (13)
5 (14)
5 (15)
5 (16)
5 (17)
5 (18)
5 (19)
5 (20)
5 (21)
5 (22)
5 (23)
5 (24)
6 (1)
6 (2)
6 (3)
6 (4)
6 (5)
6 (6)
6 (7)
6 (8)
6 (9)
6 (10)
6 (11)
6 (12)
6 (13)
6 (14)
6 (15)
6 (16)
6 (17)
6 (18)
6 (19)
6 (20)
6 (21)
6 (22)
6 (23)
6 (24)
7 (1)
7 (2)
7 (3)
7 (4)
7 (5)
7 (6)
7 (7)
7 (8)
7 (9)
7 (10)
7 (11)
7 (12)
7 (13)
7 (14)
7 (15)
7 (16)
7 (17)
7 (18)
7 (19)
7 (20)
7 (21)


7 (22)
7 (23)
7 (24)
8 (1)
8 (2)
8 (3)
8 (4)
8 (5)
8 (6)
8 (7)
8 (8)
8 (9)
8 (10)
8 (11)
8 (12)
8 (13)
8 (14)
8 (15)
8 (16)
8 (17)
8 (18)
8 (19)
8 (20)
8 (21)
8 (22)
8 (23)
8 (24)
9 (1)
9 (2)
9 (3)
9 (4)
9 (5)
9 (6)
9 (7)
9 (8)
9 (9)
9 (10)
9 (11)
9 (12)
9 (13)
9 (14)
9 (15)
9 (16)
9 (17)
9 (18)
9 (19)
9 (20)
9 (21)
9 (22)
9 (23)
9 (24)
10 (1)
10 (2)
10 (3)
10 (4)
10 (5)
10 (6)
10 (7)
10 (8)
10 (9)
10 (10)
10 (11)
10 (12)
10 (13)
10 (14)
10 (15)
10 (16)
10 (17)
10 (18)
10 (19)
10 (20)
10 (21)
10 (22)
10 (23)
10 (24)