UNIVERSITY OF NAIROBI

SCHOOL OF COMPUTING AND INFORMATICS

A Comparative Study of Minutiae Based Fingerprint Matching Algorithms

NJERU, SILAS KIVUTI

P58/61707/2010

SUPERVISOR:

Dr. ROBERT OBOKO

DECEMBER 2015

A project submitted in partial fulfilment of the requirement for the award of a degree of Masters of Science in Computer Science at the School of Computing and Informatics, University of Nairobi.

DECLARATION OF ORIGINAL WORK

I declare that this research project is my original work and has not been submitted to the University of Nairobi and any other university to the best of my knowledge for the same purpose in the same scope and area of research.

NAME: SILAS KIVUTI NJERU REG. NO: P58/61707/2010

SIGNATURE…… DATE……

This research project report is submitted for presentation with my approval as the

University Research project supervisor

NAME: DR.ROBERT OBOKO

SIGNATURE…….. DATE……..

DEDICATION

This work is dedicated to my sons Brian Murimi, Ken Mutugi and Peace Murathime for their patience and support since the beginning of my studies

Secondly, I dedicated this work to my parents who gave me the passion to advance in my knowledge. Also, this thesis is dedicated to my uncle Meshack Muturi who has been a great source of motivation and inspiration.

ACKNOWLEDGEMENTS

Foremost, I would like to express my sincere gratitude to my supervisors Dr. Robert Oboko and Ms Pauline Wambui for their continuous support throughout my research for their patience, motivation, enthusiasm, and immense knowledge. Their guidance helped me at all the time of research and writing of this thesis.

My sincere thanks also goes to Mr. James Muhati, Mr Michael Ouma, Stephen Ngeno, Ronald Chamwanda, and Abdulahi Abdi, for offering a condusive working environment and diverse ideas to help advance my research project.

ABSTRACT

The fingerprint recognition has been used widely in cross border identification, criminal investigation, access control and paternity identification among other modern identification and authentication systems. However, the complex distortions among the different fingerprint impression in real life poses a challenge in the performance of fingerprint recognition systems. Matching two fingerprints or finding duplicates in a large database of fingerprints can be difficult due to various reasons depending upon the method that is being used for matching. In the modern day technologies, various biometric identifiers e.g. iris, voice, fingerprints are considered more reliable for person recognition than traditional password or knowledge based methods because they cannot be easily misplaced, forged, or shared.

Previous research on this domain has only focused on comparing algorithms based on building confusion matrix or using false acceptance rate (FAR) and false rejects rates (FRR) which are measures of accuracy. In this research, we identified speed as another parameter in the comparative analysis. In our experiment, we compared two fingerprint matching algorithms by simulating the matching process of sampled fingerprint images. The process involved comparing each of the four (4) fingerprints from each individual with the entire set of other candidate fingerprints to identify duplicates if they exist. The output of a match comparison is either a positive match or a negative match. Based on the result of accuracy, time taken for matching, and the number of similar featured identified, the best algorithm was determined and a prototype system for de-duplication was developed. The two types of matching techniques used in this research were based on (a) matching using global orientation minutiae features and (b) matching using minutia triangulation technique. We conducted the experimental evaluation on a datasets of 100 candidates using four (4) fingerprints from each candidate. The data was sampled from a mass registration of citizen in Kenya conducted by a reputable organization.

The research revealed that fingerprint matching based on minutia triangulation algorithms performs better in terms of speed with an average of **38.32** milliseconds as compared to matching based on a combination of minutia and global orientation features with an average of **563.76** milliseconds. In terms of accuracy of matching, the algorithms based on a combination of minutia and global orientation field features performs better with an average similarity score of **0.142433** as compared to m-triplet based matching with an average similarity score of **0.004202**.

TABLE OF CONTENTS

DECLARATION OF ORIGINAL WORK ii

DEDICATION iii

ACKNOWLEDGEMENTS iv

ABSTRACT v

TABLE OF CONTENTS vi

LIST OF FIGURES ix

LIST OF TABLES x

LIST OF ABBREVIATIONS xi

CHAPTER ONE – INTRODUCTION 1

1.1 Background 1

1.2 Problem Statement 3

1.3 Research Objectives 4

1.4 Research Questions 5

1.5 Expected Outcome 5

CHAPTER TWO - LITERATURE REVIEW 6

2.1 Introduction 6

2.2 Fingerprint Features 6

2.2.1 Global Ridge Pattern (Level 1 Features) 6

2.2.2 Local Ridge Pattern (Level 2 Features) 7

2.2.3 Intra-Ridge Detail (Level 3 Features) 7

2.3 Image Pre-Processing 8

2.4 Image Feature Extraction 8

2.5 Types of Fingerprint Matching Algorithms 9

2.4.1 Artificial Neural Network Based 10

2.4.2 Correlation Based Algorithms 11

2.4.3 Fingerprint Matching Based Local and Global Structures 12

2.4.4 M-triplet Descriptor Based Matching 13

2.6 Fingerprint Recognition Process 14

2.5.1 Enrolment Stage 15
2.5.2 Feature Vector Extraction 16
2.5.3 Recognition Stage: Matching stage 16
2.5.4 Computation of Similarity Scores 16
2.7 Empirical Literature 17
2.6.1 Examples of Fingerprint Image Pre-Processing Routines 19
2.6.2 Segmentor Routine 19
2.6.3 Enhancement Routine 19
2.6.4 Ridge-Valley Orientation Detector 20
2.6.5 Feature extraction 20
2.6.6 Matching Technique 21
2.8 Performance Evaluation Of Fingerprint Recognition Algorithms 21
2.7.1 Receiver Operator Curve Analysis 21
2.7.2 False Match Rate Vs False Non-Match Rate 22
2.7.3 Comparison of Minutia points and Minutia Triangulation 23
2.7.4 Proposed Comparative Analysis 24
CHAPTER THREE - METHODOLOGY 25
3.1 Introduction 25
3.2 Requirements Analysis 26
3.2.1 Functional Requirements 26
3.2.2 Non-Functional Requirement 27
3.3 Interface Design 27
3.3.1 Main form 27
3.3.2 Matching Experiment Form 28
3.4 Data Collection and Conversion 29
3.5 Sampling Technique 30
3.6 Algorithm Implementation 30
3.7 Prototype Testing 31
3.8 Visual Match Test 32

CHAPTER FOUR – RESULTS AND DISCUSSIONS 34
4.1 Introduction 34
4.2 Overall Results 34
4.3 Analysis of Results Using Frequency Distribution Graphs 35
4.3.1 Graphs for Minutia Triangulation Algorithm (M3gl) 35
4.3.2 Graphs for Minutia and Global Orientation Algorithm 37
4.4 Discussions 39
CHAPTER FIVE - CONCLUSIONS AND RECOMMENDATIONS 40
5.1 Introduction 40
5.2 Achievements 40
5.3 Recommendations 41
5.3.1 Recommendations for Research 41
5.3.2 Recommendations for Practice 41
5.4 Future Work 41
5.5 Limitations of this Research 42
REFERENCES 43
APPENDIX I– Sample Output for M3gl Algorithm 45
APPENDIX II – Sample Output for MQYW Algorithm 53
APPENDIX 3 – Sample Codes 60

LIST OF FIGURES

Figure 1 - Global Level features....6
Figure 2 - Local Ridge Details 7
Figure 3 - Artificial Neural Network....11
Figure 4 - M-triplet Feature representation 14
Figure 5 - Operations for Biometric Recognition System....15
Figure 6 - Stages of fingerprint recognition (Nadarajah et. al 2011) 15
Figure 7 - Feature Extraction....16
Figure 8 - Image orientation field....18
Figure 9 - Flow chart of the minutiae extraction process....21
Figure 10- Confusion matrix 22
Figure 11 - MTriplet Features 23
Figure 12 - Minutiae Points....23
Figure 13 - Crossing Numbers 24
Figure 14 –Software development process flow 25
Figure 15- Requirements Analysis Process....26
Figure 16 - Main Form 27
Figure 17 - Main Menu....28
Figure 18 - Experiment Form....28
Figure 19 - Save Results Dialog Box 29
Figure 20 -Image conversion screen....30
Figure 21 -Labelling of the Fingers....31
Figure 22 - Left Hand....31
Figure 23 -Visual match of the same fingerprints....33
Figure 24-Visual Match of different fingerprint images 33
Figure 25-Minutia Triangulation - Bar Chart on Accuracy....35
Figure 26- Minutia Triangulation - Bar Chart on Time taken (ms) 36
Figure 27 -Minutia Triangulation - Bar Chart on No of similar features....36
Figure 28 -Global Orientation - Bar Chart on Accuracy....37
Figure 29 -Global Orientation - Bar Chart on Time taken (ms)....38
Figure 30 -Global Orientation - Bar Chat on No of Similar Features....38

LIST OF TABLES

Table 1 - Query Fingerprint.....32

Table 2 - Template fingerprints.....32

Table 3- Sample Matching Results using Minutia Triangulation 34

Table 4 - Sample Matching Result using Minutia and Global Orientation Feature.....34

Table 5 - Minutia Triangulation (M-Triplet) Based.....39

Table 6 -Minutia and Global Orientation Based 39

LIST OF ABBREVIATIONS

ANSI- NIST	:	American National Standards Institute – National Institute of standards and technology.
ANN	:	Artificial Neural Network
AFIS	:	Automatic Fingerprint Identification System
DPI	:	Dots Per Inch
DFT	:	Discreet Fourier Transform
FAR	:	False Accept Rate
FMR	:	False Match Rate
FRR	:	False Reject Rate
FNMR	:	False Non-Match Rate
GUI	:	Graphical user interface
ROC	:	Receiver Operating Characteristics
POC	:	Phase Only Correlation

CHAPTER ONE – INTRODUCTION

1.1 Background

Biometric recognition refers to the automatic recognition of individuals based on their physiological and/or behavioural characteristics (Dela & Grgic 2004). The use of biometrics relies on the presumption that individuals are physically and behaviourally distinctive in a number of ways. Any human physiological and/or behavioural characteristic can be used as a biometric as long as these characteristics are universal, distinct, permanent and collectable amongst a large population. Some of the physiological characteristics that have been used in biometrics recognition include face, iris, fingerprints, and voice amongst others. Biometric systems have increasingly been used in authentication and security access in banking systems, cross border identification, forensics, criminal investigations, paternity determinations, citizen registration and electronic voting systems (Pato & Millette 2010)

Use of biometrics in the registration of citizens is becoming one of the most interesting and emerging technology applications in the authentication and identification persons as compared to traditional password authentication. For instance in Kenya, biometric registration has been used for registration of voters and civil servants. This process involves collecting biometric data such as finger prints, facial scans, voice, signatures and iris for the purposes of identification. The purpose of these technologies is mainly to help in the detection and elimination of imposters as well as elimination of multiple registrations thus improving the accuracy, reliability and effectiveness of the electronic authentication.

Pato & Millette (2010), observes that biometric recognition system involves probability matching of records within a tolerance of approximation of observed biometric traits against previously collected data for a subject. Uncertainties in biometric systems arise from variations within persons e.g. changes in age, environment, disease, emotions, occupation, training and other intentional alterations. Other variations may result from the sensitivity and calibration of the sensor devices used as well as the biometric feature extraction and matching algorithm used. Therefore, biometric traits have fundamental statistical properties, distinctiveness and differing degrees of stability under natural physiological conditions and environmental challenges. In large scale applications, the underlying biological properties and distributions of biometric traits in a population are generally observed only through image filters and require probability decision making both by the automated recognition system and the human interpreters of the results. A biometric match therefore represents not a certain and definitive recognition but a probability of correct recognition. The authors also hypothesis that a fraction of results from even the best designed biometric system might be

incorrect or indeterminate. They noted that users and developers of biometric systems should recognize and take into account the limitations and constraints of biometric systems especially the probabilistic nature of underlying science, the limited knowledge regarding human distinctiveness and the numerous sources of uncertainty in biometric systems.

Pato & Millette (2010) noted that one typical assumption in the design of most biometric system has been that if the biometric features are properly collected and stored, they are sufficiently distinctive to support the application in question. In practice, where the algorithms cannot uniquely distinguish between two records within a high degree of accuracy based on a set threshold, other manual techniques are employed which includes manual investigation and verification of the records.

Generally, fingerprint-matching algorithms have two steps namely; (a) align the fingerprints and (b) find the correspondences between two fingerprints. The approach proposed by Jain et al. (1997) is capable of compensating for some of the nonlinear deformations and finding the correspondences. However, since the ridges associated with the minutiae are used to estimate the alignment parameters, the size of the templates has to be large, which takes much memory and computation, otherwise, the alignment will be inaccurate. Jiang & Yau (2005) use the local and global structures of minutiae in their approach. The local structure of a minutia describes a rotation and translation invariant feature of the minutia in its neighbourhood, and the global structure tries to determine the uniqueness of a fingerprint. The problem with this technique is that it cannot compensate for real world distortions of a 3-Dimensional elastic finger.

Besides minutiae, researchers have used other fingerprint features for matching. Saleh & Adhami (2001) proposed an approach which transforms fingerprint images into a sequence of points in the angle-curvature domain. The matching between a query fingerprint and a template fingerprint is based on the least-squares error of the Euclidean distance between corresponding points in the angle-curve domain. Jain et al. (1997) presented a filter-based algorithm, which uses a bank of Gabor filters to capture both local and global details in a fingerprint as a compact fixed length Finger Code. The authors reported that the Finger Code-based system performs better than a state-of-the-art minutiae-based system when the performance requirement of the application system does not demand a very low false acceptance rate.

1.2 Problem Statement

There is generally lack of knowledge in the industry with regard to an optimal algorithm for fingerprint enrolment and identification. Biometric systems are resource intensive in terms of processing speed and memory and thus the need to evaluate performance based on the two parameters. With the limited computing and time resources, it is important to develop reliable performance metrics for algorithms that can be applied in the selection of biometric matching algorithms. The most common measures of performance for biometric identification systems are based on accuracy of matching. A number of design factors have continued to create bottlenecks in achieving the desired performance of recognition systems. These include the lack of reliable minutia extraction algorithms, difficulty in quantitatively defining a reliable match between fingerprint images, poor image acquisition, low contrast images as well as the difficulty of reading the fingerprint for manual workers.

The quality of captured fingerprints impacts on the accuracy and performance of the biometric identification system. Poor fingerprint image quality affects the identification system because of the additional processing workload to eliminate spurious features as well as missed identification / verification of the subject. A number of other factors that affects the quality of fingerprint images include; dry fingers due to natural aging leading to light prints , Worn ridge structure due to occupation, Finer ridge structure due to demographic group e.g Male vs Female, Age group, Uncooperative or nervous subject. This can result to incorrect finger placement, Humidity / Temperature, Ambient light, unclean scanner surface and ease of use of the capture software.

Amongst the techniques that have been used to address these challenges include finger preparation like moisturizing as well as use of scanner silicon membrane /coating. New technologies have also emerged which uses touch-less scanners. Training of the operator is an important aspect to eliminate errors caused by incorrect placement of the fingerprint. Improvements should also be done on the software GUI to provide guidance to the user on how to correctly place the finger.

The application of fingerprint biometrics in voter registration and civic registration poses a challenge of accurately capturing the biometric traits of individuals as well as extracting and matching them to eliminate duplicates (de-duplication) within an acceptable margin of error. According to Pato et. al (2010), an effective biometric solution does not have to be nor can it be perfect, accurate and secure. A biometric algorithm at its core is a comparison system, taking biometric samples as input and producing a measure of similarity

as its output. This similarity or otherwise called the matching score is particular to an algorithm and is the fundamental output of the matching process. The enrolment process involves presenting the finger to the sensor (often two or more times) so that the system can record all the important and distinctive details from the fingerprint. The details captured from this process are stored as reference template in order to allow it to recognize the finger every time it is presented in the future. Image pre-processing techniques have been used by various algorithms which include image enhancement, thinning, binarization and feature extraction techniques. The effectiveness of the pre-processing techniques is a continuous research question.

The American Nation Institute of standards and technology (NIST) has carried extensive research in the area of biometrics over many years. The institute has also developed prototype algorithms for fingerprint recognition. However, there has not been any single algorithm that has been proved to be perfect in uniquely discriminating individual and thus this has remained a continuous research area.

A biometric recognition project suffers a number of challenges resulting from system specifications, design, implementation and support. For instance, most biometric systems suffer from minutiae correspondence problems and as result causing multiple false duplicate due to the low quality of fingerprint images, As a result manual verification are used to compare faces and textual data by a trained user. Most Biometric systems cannot uniquely identify individuals due to either low quality of captured images or damaged fingers.

1.3 Research Objectives

This research aims at studying two minutia based algorithms, one that is based on a combination of the global and local minutia features and another one that is based on the m-triplet minutia features. This investigation is therefore classified as a technology investigation that aims at identifying an optimal algorithm for finger print feature extraction and matching.

The overall goal of this research is to carry out an exploratory analysis of fingerprint algorithms and propose an optimal algorithm that can fulfil the requirements for fingerprint verification for biometric recognition systems. This investigation should prove that verification results are within a given acceptable tolerance. The specific objectives will include:

i. To analyse effectiveness of minutiae based fingerprint matching algorithms.
ii. To compare the performance of minutia based matching algorithms versus Minutia triangulation (M-triplet) based in terms of speed and accuracy.
iii. To develop and implement a prototype for fingerprint matching and de-duplication using a sample dataset of citizens in Kenya.

1.4 Research Questions

The following research questions have been generated based on the above research objectives.

i. How effective are minutia based fingerprint recognition algorithms?

ii. How do algorithms based on minutia compare with M-triplet based algorithms in terms of speed and accuracy?

iii. How can a fingerprint recognition system prototype be implemented based on the optimal matching algorithm?

1.5 Expected Outcome

The main outcome shall be a quantitative comparison of the performance of the minutia based and minutia triangulation (m-triplet) fingerprint matching algorithms based on a database of fingerprints obtained during citizen registration in Kenya.

CHAPTER TWO - LITERATURE REVIEW

2.1 Introduction

In this chapter, we will discuss those algorithms concerned with finger print recognition but specifically algorithms in the category of Minutiae based and Probabilistic Neural Network. In this chapter, we shall study the different fingerprint classification that can be used in the recognition process.

A fingerprint can be defined as an impression of the epidermal ridges and valleys at the surface of a human fingertip. These impressions can be classified into three categories, namely, Level 1 (Global ridge patterns), Level 2 (Local ridge patterns- minutiae points) and Level 3 (Intra-ridge details - pores and ridge shapes) that are used for recognition purposes. The finger print recognition systems uses automated method and algorithms to verify a match of a candidate finger print against a database of fingerprint templates. Most fingerprint systems can be categorize as either verification (1:N) or identification (1:1) systems where verification process either accepts or rejects a user identity based on a match against an existing fingerprint database, whereas identification establishes whether the user is who he/she claims to be. Chaohong (2007) postulates that most fingerprint recognition algorithms are designed to minimize two types of errors namely the False Acceptance Rate (FAR) and False Reject Rate (FRR). FAR and FRR refer to errors in the matching process and are closely related to the more frequently reported false acceptance rate (FMR) and the false rejection rate (FNMR). FAR and FRR refer to results at a broader system level and include failures arising from additional factors, such as the inability to acquire a sample. Although there are several methods for detecting minutiae, the technical problem of feature extraction is still an active research problem.

2.2 Fingerprint Features

2.2.1 Global Ridge Pattern (Level 1 Features)

These are based on the ridge flows of fingerprints that create particular patterns, such as shown in Figure 1 below. (a) Left-loop, (b) Right-loop, (c) Whorl, (d) Arch, and (e) Tented-arch

Figure 1 - Global Level features

An example of algorithm that uses global ridge detail is the PCASYS developed by the NIST. This Algorithm is a neural-network based fingerprint classification system, which categorized a fingerprint image into the class of arch, left or right loop, scar, tented arch, or whorl.

2.2.2 Local Ridge Pattern (Level 2 Features)

These are the patterns that form on the friction ridges of the fingerprints and they do not run evenly across our fingers, hands, toes and feet but rather, they display a number of characteristics known as minutiae. The most common minutiae are;

a) Ridge endings: These are ridges that end abruptly.
b) Ridge bifurcations: This is a single ridge that divides into two.
c) Lake or enclosure : This is a single ridge that bifurcates and reunites shortly afterwards to continue as a single ridge;
d) Short ridge, island or independent ridge: This is a ridge that commences, travels a short distance and then ends.
e) Dot : This is an independent ridge with approximately equal length and width;
f) spur : This is a bifurcation with a short ridge branching off a longer ridge; and
g) Crossover or bridge: This is a short ridge that runs between two parallel ridges.

Figure 2 below illustrates level 2 features of a fingerprint image.

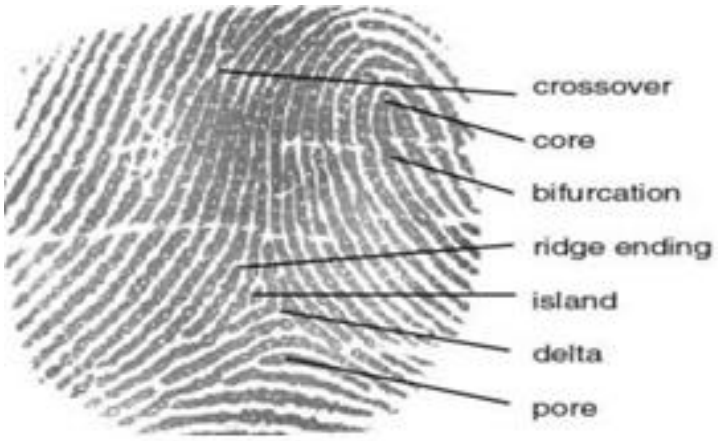


Figure 2 - Local Ridge Details

2.2.3 Intra-Ridge Detail (Level 3 Features)

These are based on the very-fine level details of the finger print. The most important feature is finger sweet pore, which can be observed using a high resolution sensor (1000 dpi). Level 3 features are still under research and development stage as it requires high-resolution image capturing to extract and process.

2.3 Image Pre-Processing

The quality of a fingerprint image is determined by many factors which may be difficult to control; therefore fingerprint systems must be able to handle images in medium and low quality. Feature enhancement routines are applied on an image to improve on the quality of the fingerprint features for effective recognition.

Some of the common image enhancements techniques are based on computation of the forward two-dimensional fast Fourier transform (FFT) to convert the data from its original (spatial) representation to a frequency representation. Thereafter, a nonlinear function is applied that increases the power of useful information (the overall pattern, and in particular the orientation, of the ridges and valleys) relative to noise. Finally, the backward 2-d FFT is done to return the enhanced data to a spatial representation before snipping out the middle 16×16 pixels and installing them into the output image (Kenneth et.al 2007)..

These enhancement routine uses localized FFT filter techniques on the input image thus increasing the accuracy of the resulting classifier. The nonlinear function applied to the frequency-domain representation of the square of pixels has the effect of increasing the relative strength of those frequencies that were already dominant. The dominant frequencies correspond to the ridges and valleys in most cases. So the enhancer routine strengthens the important aspects of the image at the expense of noise such as small details of the ridges, breaks in the ridges, and ink spots in the valleys. (Kenneth et al 2007)

In this research, a uniform image enhancement technique is applied on each of the fingerprint images in order to improve on the quality of the finger print recognition methods. A Gaussian blur filter is applied to the image to remove noise and extra details thus smooth the overall shape of the image. This helps in connecting the falsely broken points on ridges as well as to remove false connections between ridges. Fingerprint Image binarization is applied to transform the Grayscale fingerprint image to a 1-bit image with 0- value for ridges and 1-value for furrows. This operation causes the ridges in the fingerprint to be highlighted with black colour while furrows are highlighted with white.

2.4 Image Feature Extraction

According to Ratha et al (1995), in a good rolled fingerprint image there are about 70 to 80 minutiae points while in a latent fingerprint image the number of minutiae is much less (20 to 30). The ratha1995minutiaextractor is the basic feature extractor for the minutia classification algorithm. It uses a pixel – alignment technique for fingerprint orientation field detection. The algorithm detects, at each pixel location of the fingerprint image, the local orientation of the ridges and valleys of the finger surface, and produces an array of regional averages of

these orientations. The routine is based on the ridge-valley fingerprint binarizer that reduces a grayscale fingerprint image to a binary (black and white only) image.

Typically, the pixel–alignment-based method computes the local ridge orientation of each pixel on the basis of the neighbouring pixel alignments with respect to a fixed number of reference orientations. Differentiation (fluctuation) of neighbouring pixels grey level values is expected to be the smallest along the local ridge orientation and the largest along its orthogonal orientation. The accuracy of the estimated orientation in the pixel-alignment-based method is limited because to the fixed number of reference orientations (Kenneth et al 2007).

The feature extraction technique involves the minutia extractor, orientation image extractor and the skeleton image extractor proposed by Ratha et al (1995). In this algorithm, the feature detection and extraction techniques uses convolution filters to detect and extract features from the input and template images. Using the convolution filters, the fingerprint features are defined by a 5X5 matrix of the grayscale image. The results of the convolution matrix are divided by a bias value of 40 in order to keep the pixel values within the 0-255 range.

2.5 Types of Fingerprint Matching Algorithms

Fingerprint matching can be categorized as Neural Network based, Minutiae based or Correlation based. Neural Network matching is a pattern based matching algorithm which uses graphical comparison of the entire fingerprint image as opposed to the individual minutiae points. Some matching techniques uses global level features and while others combine both global and local features. The characteristics that are used include the ridge thickness, curvature, or density. A pattern-based algorithm is independent of the number of minutiae points in a fingerprint as well as independent of the size of the finger print sensor. Compared to other algorithms, pattern-based algorithms are not affected by the quality of the fingerprint image. However, Qi et. al (2005), observed that minutiae-based methods perform better than correlation-based due to their uniqueness, stability, speed of processing and memory requirements despite the fact that they may be affected by rotation, translation, deformation of the fingerprints as well as location and direction of the detected minutiae or presence of spurious minutiae. Therefore minutia features are considered to be more reliable and robust.

Minutia based matching consists of finding the best alignment between the extracted and stored template minutia and the minutiae from the subject finger print. On the other hand, Pattern based algorithms are based on scanning the overall fingerprint global features i.e. the loop, whorl or the arch patterns. The main problem in minutiae extraction methods is that

minutiae in the skeleton image do not always correspond with true minutiae in the fingerprint image because of false minutiae extracted as a result of undesired spikes, breaks, and holes. For this reason, time-consuming pre-processing algorithms are required prior to the matching stage.

2.4.1 Artificial Neural Network Based

A Artificial neural network (ANN) based fingerprint pattern classification algorithms are designed to automatically categorize a fingerprint image based on the arch, left loop, right loop, scar, or whorl. By first classifying a fingerprint according to its class reduces the number of candidate searches required to determine if a fingerprint matches to the file prints. This improves the computation efficiency of the matching process by partitioning the file fingerprints based on classification thus greatly reducing the number of comparisons that must be performed by the minutiae-matcher (Kenneth et.al 2007). The ANN system performs all the processes of image segmentation, enhancement, feature extraction, registration, dimensionality reduction and classification. At the classification stage, the Artificial Neural network traces and analyses ridges and creates the template (hypothesized class).

The basic method used by the fingerprint classifier consists of, first, extracting from the fingerprint to be classified an array (a two-dimensional grid in this case) of the local orientations of the fingerprint's ridges and valleys. Second, comparing that orientation array with similar arrays made from prototype fingerprints ahead of time. The comparisons are performed between low-dimensional feature vectors made from the orientation arrays, rather than using the arrays directly.

The ANN algorithm classifies an input feature vector by computing the Gaussian kernel functions values which are organized into a multi-layered feed forward network with four layers namely Input layer, Hidden layer, Pattern layer/Summation layer and the Output layer as shown below. The output classification of the fingerprint is computed by applying a transfer function f(x, y) to the weighted sums of the input vector values from the input image. To work with the fingerprint recognition system using artificial neural network requires training of the neurons from a training set of data and calculating the output of the network by applying some random weights until the error between the network output and the desired output is minimal.

Figure 3 below illustrates an Artificial Neural Network.

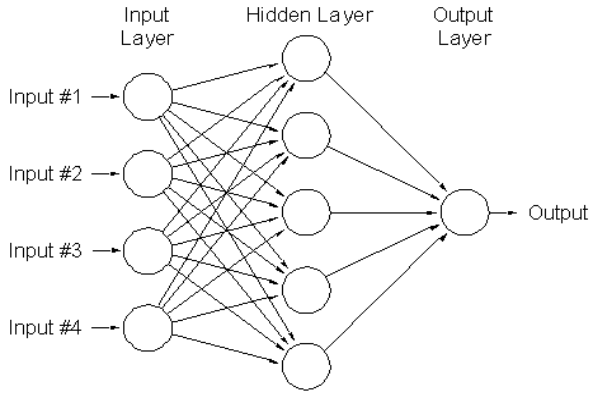


Figure 3 - Artificial Neural Network

The artificial neural network (ANN) uses a set of training datasets to build a model that can be able to classify any input variable into its target class. Each Training data point corresponds to a Gaussian Function.

$$z = f(x,y) = e^{\frac{-((x-x_o)^2+(y-y_o)^2)}{2\delta^2}} \quad \ldots\ldots\ldots (1)$$

For multiple inputs values, the equation below applies to calculate the category values.

$$z = Max\{f(x,y)\} = \sum_{i=1}^{n} e^{\frac{-\left((x-x_i)^2+(y-y_i)^2\right)}{2\delta^2}} \quad \ldots\ldots\ldots (2)$$

The algorithm picks a maximum value of the category unit from this function which becomes the corresponding output value for the classification. This algorithm requires extensive training.

2.4.2 Correlation Based Algorithms

Texture correlation and convolution techniques are also used in image processing and can also be applied to finger print matching. The phase only correlation function (POC) uses the phase spectra of the finger print images and computes the Discreet Fourier Transforms (DFT) of two finger print images. The phase spectrum transforms an image into its frequency domain representation. When two images are similar, their POC function gives a distinct sharp peak, but when two images are not similar, the peak drop significantly. Research conducted by Koishi et al (2004) noted that phase correlation techniques are not influenced by image shift and brightness change and it is highly robust against noise. Texture features have been have also been applied to fingerprint matching where the finger print is tilled into cells and a bank of Gabor filters are combine with each cell and the variance of the energies of the Gabor filer responses in each cell is used as a feature vector. These techniques are computationally expensive, although it has been suggested that local correlation and correlation in Fourier domain can improve efficiency (Roli et al 2011)

Correlation based matching uses the grey level information of the fingerprint image since it contains much richer, discriminatory information than only the minutiae locations. This takes into account the level 3 features as well as other fingerprint features. In correlation based techniques, two fingerprint images are superimposed and the correlation between corresponding pixels is computed for different alignments.

2.4.3 Fingerprint Matching Based Local and Global Structures

Jian & Yau (2000) proposed an algorithm based on local and global features e.g. minutia type, coordinates, and the orientation angle to compare the query and the template fingerprint. The algorithm computes the Euclidian distances between the feature vectors in order to obtain the correct minutiae correspondence. The final matching score in these algorithms involves measuring both the number of matching minutiae pairs and the similarity degree of two orientation fields thus reducing the false rejection rate as well as false acceptance rate as illustrated by Jian & Yau (2000). This method takes advantage of more information than traditional minutiae based method. By combining the Local structures and the fingerprint orientation field, this algorithm improves the minutiae correspondence.

A minutia point M detected from a fingerprint is described by the feature vector $f(x, y, \omega)$, Where (x, y) are the coordinates of the minutiae points, ω is the local ridge orientation direction of the fingerprint ridge in the range [$\pi/2$, $\pi/2$] or [0, π]. To measure the difference between two ridge directions, ω_1 and ω_2 the function $d\ (\omega_1, \omega_2)$ is given as below;

$$d(\omega_1, \omega_2) - \begin{cases} \omega_1 - \omega_2 & \text{if} -\frac{\pi}{2} < (\omega_1 - \omega_2) < \frac{\pi}{2} \\ \omega_1 - \omega_2 + \pi & \text{if} -\frac{\pi}{2} < (\omega_1 - \omega_2) < -\pi/2 \\ \omega_1 - \omega_2 - \pi & \text{if} \frac{\pi}{2} < (\omega_1 - \omega_2) < \pi \end{cases} \quad \ldots\ldots\ldots (3)$$

Given a minutiae point M with orientation ω, a minutiae structure is defined as follows;

Let $\theta_1 = \omega,\ \theta_2 = \theta_1 + \frac{2\pi}{3},\ \theta_3 = \theta_2 + 2\pi/3$ ..... (4)

Using the above minutia feature vectors, Jiang & Yau (2000) developed an algorithm that receives as the input two minutia lists and two orientation fields captured from two fingerprint impressions and delivers a matching score that expresses the degree of similarity between the two fingerprints. The value of the similarity level between minutiae (b_1, b_2) is obtained by maximizing the similarity level $s(b_1, b_2) = \max_{i,j}(s\ (i, j))$, where i and j are the minutiae points of the input and template fingerprint. The algorithm applies two

thresholds namely GlobalAngleThr and the GlobalDistThr. The GlobalAngleThr is used to compare angles in the global minutia matching step while the GlobalDistThr is used to compare minutia distances in the global minutia matching step.

2.4.4 M-triplet Descriptor Based Matching

Medina-Perez et. al (2011) Proposes an M-triplet feature using three triangular minutiae points where the ridge continuity breaks and they are typically represented as (x; y; Θ); where (x; y) represent the two dimensional point coordinates, and Θ the ridge direction at that point. Minutiae detection algorithm takes a fingerprint image and locates features in the ridges and furrows of the skin. Points are detected where ridges end or split, and their location, type, orientation, and quality are stored and used for search. There are 100 minutiae on a typical ten-print, and matching takes place on these points rather than the 250,000 pixels in the fingerprint image.

According to Medina-Perez et. al (2011), Minutiae triplets are local structures represented by three minutiae in the neighbourhood. Minutiae matching techniques aims at finding the minutiae correspondence (number of common minutiae points) between the input and the query fingerprints. In this algorithm, the number of matching minutiae points can be maximized if a proper alignment between the query and template fingerprints can be found (Kumar & Begum 2013).

The relative transformation between the Query image and the Template image poses a challenge known as correspondence problem. This causes ambiguity since each minutiae of one finger print can be matched onto any minutiae of the other fingerprint. To reduce the ambiguity, additional information is added called the minutiae descriptor information. This information helps to quickly establish the minutiae correspondence. A simple and accurate descriptor is based on Minutiae triplet (m-triplet). Some of the quality parameters of the minutiae triplet based matching include:

i. The minutiae order in the triplet does not affect the correspondence and thus the algorithm finds the correct correspondence when matching.
ii. The algorithm does not match a triplet with its reflected versions.
iii. In order to find similar triplets, the algorithm takes into account the directions of the minutiae relative to the sides of the triangles formed by the triplets.

An m-triplet is defined as a tuple with the following components (Miguel, et. al 2012);

- Minutiae $p_i \epsilon P$ which are clockwise starting at P_i
- $d_{i\epsilon 1\ldots 3}$ where d_i is the Euclidean distance between the minutiae different that p_i

- $d_{max}, d_{mid}, and\ d_{min}$ Which are the maximum, middle, and minimum distance values.
- $\alpha_{i\in 1\ldots 6}$ which are the angles $ad_{2\pi}(ang(p,q),\theta)$ required to rotate the direction θ of a minutia to superimpose it to the vectors associated with the other two minutia in the triplet.
- $\beta_i = ad_{2\pi}(\theta_i,\theta_k)$is the angle required to rotate the direction of the minutia p_k in order to superimpose it to the direction of the minutia p_j

  Figure 4 below represents the m-triplet feature

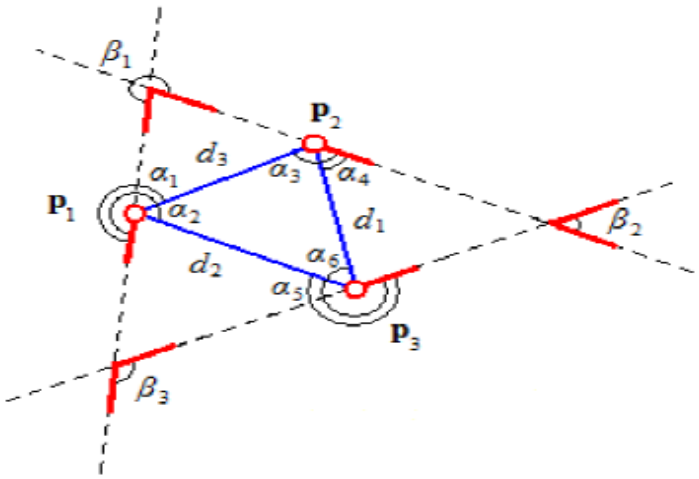


Figure 4 - M-triplet Feature representation

The M3gl developed by Medina P. et. al (2012) is based on the M-triplet feature such that given a fingerprint described by the features set P we can compute the M-triplet as follows. For each $p \in P$, we find its nearest minutiae in P and build all m-triplets that include p and two of its nearest minutiae, discarding duplicates.

2.6 Fingerprint Recognition Process

The basic operations performed by a general biometric system are the capture and storage of enrolment (reference) biometric samples and the capture of new biometric samples and their comparison with corresponding reference samples (matching). The primary component is the image capture process. In this process, the finger print sensor collects biometric data and stores into a reference database. The reference database contains previously enrolled fingerprint biometric data. The second component called the matcher, compares presented sample data to the reference data in order to make a recognition decision (Joseph et al 2010).

Figure 5 below represents the fingerprint recognition process

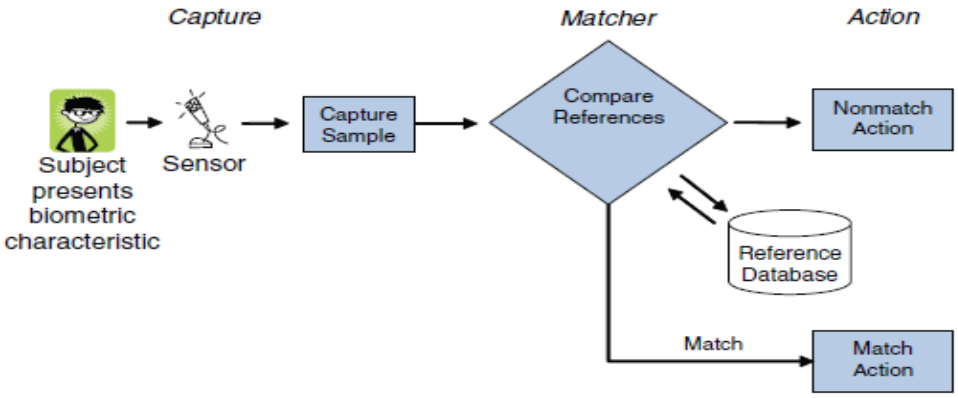


Figure 5 - Operations for Biometric Recognition System

All fingerprint recognition systems follows a two stage process which mainly consist of enrolment and matching stages. Each stage consists of several sub-stages. These can be illustrated as shown in figure 6 below;

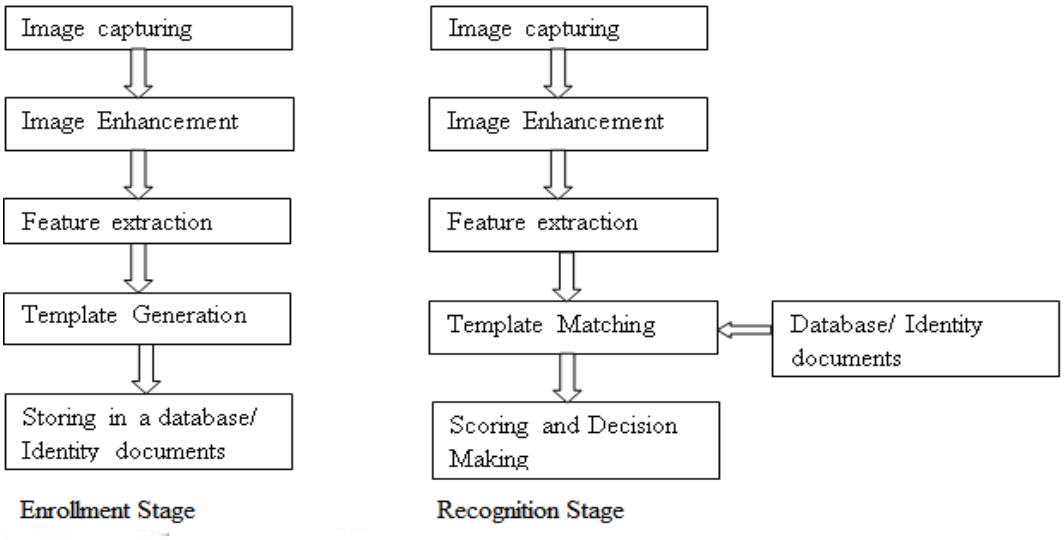


Figure 6 - Stages of fingerprint recognition (Nadarajah et. al 2011)

2.5.1 Enrolment Stage

This is the stage where each individual fingerprint is enrolled with a unique identifier. The image of the fingerprint is captured using one of the capturing techniques like optical, capacitive and Radio Frequency (RF). The captured image is then processed using image enhancement techniques. The resulting image is in a binary form. The Global and Local (level 1 and 2) features are extracted at this stage. Upon completion of the extraction process, templates are generated and stored in a database for the use in matching process for 1: N matching or in an identity document such as identity card or passport for 1:1 matching. During this stage, the image Quality algorithm analyses a fingerprint image and assigns a quality value of 1– 5 (Highest to lowest).

2.5.2 Feature Vector Extraction

The feature extraction process entails reading and codifying each of the minutiae features and generating the feature vector consisting of the x-coordinates, y-coordinates and the angular distance as shown in Figure 7 below. The minutia points are normally extracted during enrolment as well as during authentication. In industry applications, these features are codified and stored in a reference database for future recognition.

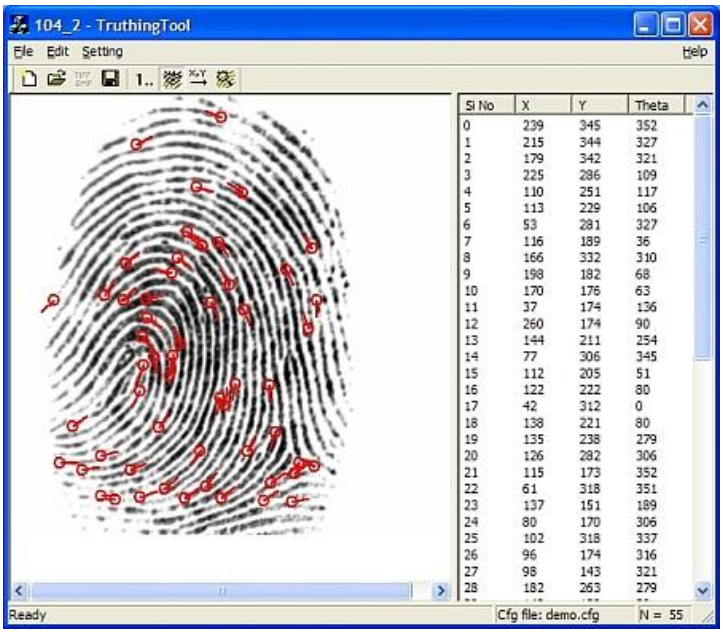


Figure 7 - Feature Extraction

2.5.3 Recognition Stage: Matching stage

This stage entails matching the extracted features against either the templates stored in data base for 1: N matching or the single template stored in the identity document (e.g.: identity card or passport) for 1:1 Matching. Depending upon the criteria used, the results from the template matching are scored and final decision is arrived to accept or reject the subject as the subject that claimed to be.

2.5.4 Computation of Similarity Scores

A similarity score is used to measure the degree with which minutiae points from fingerprints of the same finger and of different fingers can be discriminated. Minutia points are represented on the Cartesian coordinate system x, y, Θ. To compute a similarity score, the algorithm computes the Euclidian distances between the two points of a fingerprint features. Given two fingerprint images with 'T' and 'Q' identified minutiae points respectively (where T need not be equal to Q), this algorithm outputs the 'M' common minutiae points in both the images. Effectively, if T represents the set of minutiae points in image 1 and Q represents the set of minutiae points in image 2, M would be the intersection of T and Q ($M = T \cap Q$). A fingerprint matcher takes two fingerprints vectors, T_i and Q_j and produces similarity

measurement $S(T_i,Q_i)$ which is normalized in the interval [0, 1]. If the value of the matching score is close to 1, then the matcher has a higher confidence of similarity. For instance, Let the number of minutiae in T and Q be m and n respectively

$$T=m_1, m_2 \ldots\ldots m_m, \quad mi = x_i, y_i, \emptyset_i \quad i=1,\ldots m \qquad \ldots\ldots\ldots (5)$$
$$Q= m'_1, m'_2 \ldots\ldots m'_m, \quad m'j = x'_j, y'_j, \emptyset_i \quad j=1,\ldots m$$

This category of algorithms computes the Euclidean distances between the pairs of minutiae (Feature vectors). The outputs of each comparison is either a "match" or a "non-match". In Minutiae based algorithms, a minutiae m_i in T and m_j in Q are considered matching if the following conditions are satisfied. This can also be written according to the equation below using the spatial distance (sd) and direction distance (dd)

$$sd(m'_j, m_i) = \sqrt{(x^i_j - x_i)^2 + (y'_j - y_i)^2} \leq r_o \qquad \ldots\ldots\ldots (6)$$
$$dd(m'_j, m_i) = \min(|\theta_j - \theta_i|, 360 - |\theta_j - \theta_i|) \leq \emptyset o$$

where r_o and $\emptyset_o$ are the parameters of tolerance required to compensate for errors.

Minutia based fingerprint matching system usually returns the number of matched minutia on both the query and reference fingerprint and uses it to generate similarity scores in the rage between [0….1]. More matched minutiae will always yield a higher similarity score and thus when the number of minutiae on both the fingerprints is large, then we can confidently distinguish the genuine and impostor fingerprints using the number of matched minutiae.

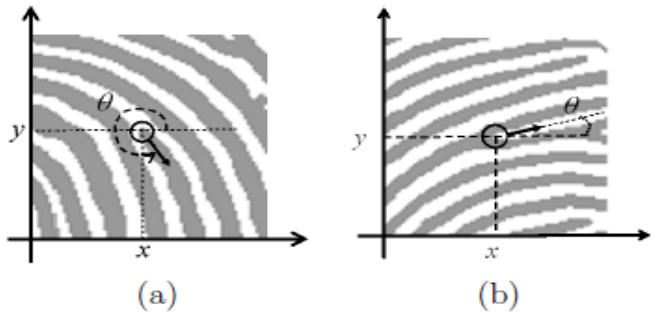
2.7 Empirical Literature

In General terms, Fingerprint matching can be classified as either Minutiae – based or Correlation/Pattern Methods. Pattern based matching algorithms uses graphical comparison of the entire fingerprint image as opposed to the individual minutiae points. The characteristics that are used include the ridge thickness, curvature, or density. A pattern-based algorithm is independent of the number of minutiae points in a fingerprint as well as independent of the size of the finger print sensor. Compared to other algorithms, pattern-based algorithms are not affected by the quality of the fingerprint image. However, Qi et. al (2005) observed that minutiae-based methods perform better than correlation-based despite the fact that they may be affected by rotation, translation, deformation of the fingerprints as well as location, direction of the detected minutiae or presence of spurious minutiae.

Minutia based matching consists of finding the best alignment between the extracted and stored template minutia and the minutiae from the subject finger print. On the other hand, Pattern based algorithms are based on scanning the overall fingerprint global features i.e. the

loop, whorl or the arch patterns. Pattern based techniques are considered to be more stable and robust to fingerprint orientation, quality and do not require extensive pre-processing or enhancements.

The American National Standards Institute – National Institute of standards and technology (ANSI- NIST) proposed a minutia-based fingerprint representation which included location and orientation. In this case, minutia orientation is defined as the direction of the underlying ridge at the minutia location. These characteristics are then represented in the Cartesian coordinates system of x, y, Θ of the miniature datasets where x- is the x- coordinates, y is the y-coordinates and Θ is the angular orientation of the miniature. The simplest and most commonly used techniques are based on segmentation, Image Orientation field estimation, binarization and ridge thinning. Figure 8illustrates the Image Orientation field estimation technique,



As shown on figure 8, a ridge ending minutia: (x,y) are the minutia coordinates; Θ is the minutia's orientation; (b) A ridge bifurcation minutia: (x,y) are the minutia coordinates; Θ is the minutia's orientation.

Figure 8 - Image orientation field

The Segmentor takes an input image that is 512x480 pixels and cuts a rectangular region to produce an output image rectangle image. The sides of the rectangle that is cut out are not necessarily parallel to the corresponding sides of the original image. The Segmentor attempts to position its cut rectangle on the impression made by the first joint of the finger. It also attempts to define the rotation angle of the cut rectangle and remove any rotation that the finger impression had to start with.

The other enhancement techniques applied on an image involves picking up parts of the image in form of input squares and performing the forward two-dimensional fast Fourier transform (FFT) to convert the data from its original (spatial) representation to a Frequency representation. Next, a nonlinear function is applied that increases the power of useful information (the overall pattern, and in particular the orientation, of the ridges and valleys) relative to noise. This enhances the segmented image before extracting the orientation features, thus increasing the accuracy of the resulting classification.

2.6.1 Examples of Fingerprint Image Pre-Processing Routines

2.6.2 Segmentor Routine

The Segmentor reads the input fingerprint image (512x480 at 500 pixels per inch) and produces an image that is 512×480 pixels in size. This involves cutting a rectangular region out of the input image. The sides of the rectangle that is cut out are not necessarily parallel to the corresponding sides of the original image. The routine attempts to position its cut rectangle on the impression made by the first joint of the finger. It also attempts to define the rotation angle of the cut rectangle and remove any rotation that the finger impression had to start with. Cutting out this smaller rectangle reduces the amount of data that has to undergo subsequent processing. Removing rotation helps in removing a source of variation between prints of the same class.

2.6.3 Enhancement Routine

The enhancement routine goes through the image and snips out a sequence of squares each of size 32×32 pixels, with the snipping positions spaced 16 pixels apart in each dimension to produce overlapping. Each input square undergoes a process that produces an enhanced version of its middle 16×16 pixels, and this smaller square is installed into the output image in a non-overlapping fashion relative to other output squares.

The enhancement of an input square is done by first performing the forward two-dimensional fast Fourier transform (FFT) to convert the data from its original (spatial) representation to a frequency representation. Next, a nonlinear function is applied that increases the power of useful information (the overall pattern, and in particular the orientation, of the ridges and valleys) relative to noise. Finally, the backward 2-d FFT is done to return the enhanced data to a spatial representation before snipping out the middle 16×16 pixels and installing them into the output image (Kenneth et.al 2007)..

The Enhancement routine uses localized FFT filter techniques on the segmented image thus increasing the accuracy of the resulting classifier. The nonlinear function applied to the frequency-domain representation of the square of pixels has the effect of increasing the relative strength of those frequencies that were already dominant. The dominant frequencies correspond to the ridges and valleys in most cases. So the enhancer strengthens the important aspects of the image at the expense of noise such as small details of the ridges, breaks in the ridges, and ink spots in the valleys. (Kenneth et.al 2007)

Other image enhancement technique that are applied on each of the fingerprint images to improve on the quality of the finger print recognition include Histogram equalization technique which is applied to expand the pixel value distribution of an image so as to increase the perception information.

A Gaussian filter is also applied to the image template to remove noise and extra details from the original image thus smoothen the overall shape of the image. This helps in connecting the falsely broken points on ridges as well as to remove false connections between ridges. Other technique involves use of Fingerprint Image binarization which is applied to transform the 8- bit Gray fingerprint image to a 1-bit image with 0- value for ridges and 1-value for furrows. This operation causes the ridges in the fingerprint to be highlighted with black colour while furrows are white.

2.6.4 Ridge-Valley Orientation Detector

This uses a pixel – alignment technique for fingerprint orientation field detection. The algorithm detects, at each pixel location of the fingerprint image, the local orientation of the ridges and valleys of the finger surface, and produces an array of regional averages of these orientations. The routine is based on the ridge-valley fingerprint binarizer that reduces a grayscale fingerprint image to a binary (black and white only) image.

Typically, the pixel–alignment-based method computes the local ridge orientation of each pixel on the basis of the neighboring pixel alignments with respect to a fixed number of reference orientations. Differentiation (fluctuation) of neighboring pixels grey level values is expected to be the smallest along the local ridge orientation and the largest along its orthogonal orientation. The accuracy of the estimated orientation in the pixel-alignment-based method is limited because to the fixed number of reference orientations (Kenneth et.al 2007).

2.6.5 Feature extraction

The feature extraction technique involves performing image thinning to eliminate redundant pixels of ridges until each ridge is a single pixel. Pruning and ridge filling operation on the thinned image can also be applied to remove false minutiae. Feature detection and extraction techniques are then applied on the image. A reference database is then created from the extracted features.

The figure 9 below represents a schematic representation of the process of feature extraction.

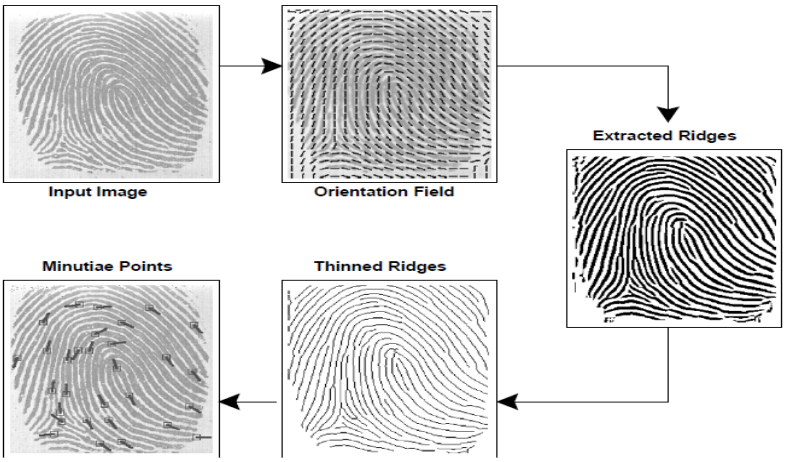


Figure 9 - Flow chart of the minutiae extraction process

2.6.6 Matching Technique

Fingerprint matching is achieved by minutia matching of the point pattern where features associated with each point pattern and inter-point distances are used to reduce the search paths.. Minutia matching processes are normally decomposed into two stages mainly (a) pre-processing stage where enhancements and transformations such as translation, rotation and scaling parameters between input minutia pattern and a template minutia are first estimated; the input minutia is then aligned with the template minutia pattern according to the estimated parameters; and (b) matching stage, where both the input and the template are converted to polygons in polar coordinate system and a matching algorithm is used to match the polygons.

In pattern based approach, the graphical center of the fingerprint image is located, then the image is cropped a fixed distance around this center point. The cropped region is then stored for subsequent matching.

2.8 Performance Evaluation Of Fingerprint Recognition Algorithms

2.7.1 Receiver Operator Curve Analysis

One of the techniques for evaluation of the performance of the algorithms that has been widely research on has been based on the receiver operating characteristics (ROC) graph. It is a graphical plot that illustrates the performance of a binary classifier system as its discrimination threshold is varied. The ROC graph is used for visualizing, organizing and selecting classifiers based on their performance. A classification model based on mapping of instances to some target class is used. If the instance is positive and it is classified as positive, it is counted as a true positive; if it is classified as negative, it is counted as a false negative. If the instance is negative and it is classified as negative, it is counted as a true negative; if it is classified as positive, it is counted as a false positive.

Given a classifier and a set of instances (the test set), a two-by-two confusion matrix is constructed as shown in figure 10 below;

		True Class	
		P	**N**
Hypothesis Class	**Y**	True Positive	False Positive
	N	False Negative	True Negative

Figure 10- Confusion matrix

From the above matrix, common metrics are calculated using Equation (7) and (8) below

$$\text{True Positive Rate} = \frac{\text{Positives correctly classified}}{\text{Total Positives}} \quad \ldots\ldots\ldots (7)$$

Similarly

$$\text{False Positive Rate} = \frac{\text{Negatives incorrectly classified}}{\text{Total Negatives}} \quad \ldots\ldots\ldots (8)$$

Thus the ROC graph is a two-dimensional graph in which **tp** rate is plotted on the Y axis and **fp** rate is plotted on the X axis. From the graph therefore, we can depict relative trade-offs between benefits (true positives) and costs (false positives). It is important to note that this technique measures performance in terms of accuracy but does not evaluate speed.

2.7.2 False Match Rate Vs False Non-Match Rate

A second technique for evaluation of the performance of the algorithms is based on the false match rate (FMR) versus the false non-match rates (FNMR). The FMR is the number of impostor comparisons with scores higher than the threshold divided by the total number of impostor comparisons computed as below;

FMR (n) = (Number of successful impostor attempts against a person n) / (Number of all impostor attempts against a person n)

Similary, The FNMR is the number of genuine comparisons with scores lower than the threshold divided by the total number of genuine comparisons computed as below

FNMR (n)= (Number of rejected verification attempts for a qualified person n) / (Number of all verification attempts for a qualified person n)

2.7.3 Comparison of Minutia points and Minutia Triangulation

The figure below visualizes and compares the two underlying techniques for fingerprint feature representation. **Figure 11** represents minutiae points while **figure 12** represents minutiae triangulation features.

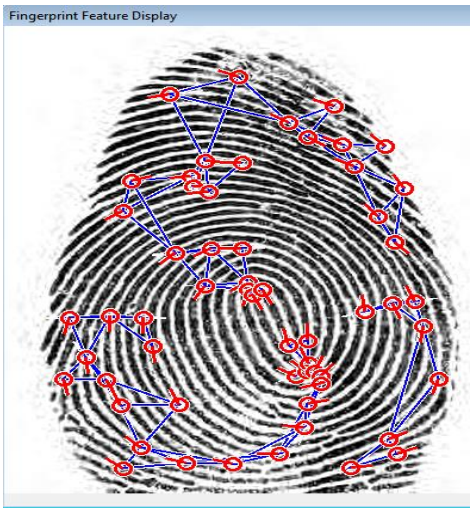


Figure 11 - MTriplet Features

Figure 12 - Minutiae Points

Minutiae matching techniques aims at finding the minutiae correspondence (number of common minutiae points) between the input and the query fingerprints. Given two fingerprint images with 'T' and 'Q' identified minutiae points respectively (where T need not be equal to Q), the algorithms outputs the '*M*' common minutiae points in both the images. Effectively, if *T* represents the set of minutiae points in image 1 and *Q* represents the set of minutiae points in image 2, *M* would be the intersection of T and Q ($M = T \cap Q$).

A classification technique categorizes the fingerprint images according to the Henry's classification scheme. This consists of the five commonly used classes namely Arch, Tented arch, Left loop, Right loop and Whorl. The commonly used method of minutiae extraction is called Crossing Number (CN). In this method, the minutiae points are determined by scanning the local neighbourhood of each pixel in the ridge thinned image using a 3x3 window.

The CN value is them computed as shown in equation 9 below which is defined as half the sum of the differences of the pairs of neighbouring pixels P_i and $P_{i+1,}$

Where

$$CN(x,y) = \frac{1}{2}\sum_{n=1}^{8} |p_i - p_{i+1}|, p_1 = p_9 \qquad \text{..... (9)}$$

The results of the formula can be summarized as shown in figure 13 below

CN	PROPERTY
0	Isolated point
1	Ridge Ending
2	Continuous ridge
3	Bifurcation
4	Crossing

Figure 13 - Crossing Numbers

The main problem, in this minutiae extraction method is that minutiae in the skeleton image do not always correspond with true minutiae in the fingerprint image because of false minutiae extracted as a result of undesired spikes, breaks, and holes. For this reason, time-consuming enhancement algorithms are required prior to thinning stage. An example of a Minutiae matching algorithm developed by the American NIST is the MIDTCT.

2.7.4 Proposed Comparative Analysis

The key aspect of the efficiency of the algorithm is measured in terms of speed of execution and accuracy of comparison. Generally, trade-offs are made across all of these measures to achieve the best-performing system consistent with operational and budgetary needs. For example, recognition error rates might be improved by using a better but more time-consuming enrolment process; however, the time added to the enrolment process could result in queues (with loss of user acceptance) and unacceptable costs. Previous comparative analysis of fingerprint carried out by Kumar & Begum (2013) compared Minutiae based matching and distance based ratio matching and observed that minutia matching performed best in terms of time and memory requirements. Other research done by Qi et. al (2004) evaluated the performance of minutiae based matching using an ROC curve. Most of the researched comparative analysis techniques have only been based of accuracy as a measure of performance.

The proposed analysis is based on combination of the speed of matching, accuracy of the algorithm and the number of features identified. The speed of matching as a performance measure has a direct impact on the speed of identification of individuals. The accuracy is evaluated using the similarity scores and the number of common fingerprint features in two fingerprints that are matched by an algorithm. From the values obtained in the experiment, descriptive statistical analysis of the similarity scores, time taken, and the number of features is performed.

CHAPTER THREE - METHODOLOGY

3.1 Introduction

This section describes an exploratory research for comparison of the performance of two fingerprint matching algorithms; one that is based on minutiae triangulation features and another based on a combination of minutia points and global orientation features. The research is conducted on a Microsoft Visual Studio environment using C# using fingerprint images obtained from a citizen registration exercise conducted in Kenya by a reputable institution. Minutiae based algorithms compare several minutia points from an original image stored in a template with those extracted from a candidate finger print. Minutiae based algorithms require extensive pre-processing and image enhancement techniques in order to improve on the degree of extraction and perception of the minutia as well as to remove spurious or false minutiae. The pre- processing mainly involves linearization and thinning techniques that require a lot of processing power and time.

In this research, we aimed at carrying out an exploratory analysis of those algorithms that are based of minutia and global orientation features namely MJY (Jian & Yau 2000) and MQYW (Medina-Pérez, et al 2012) against those that are based on minutia triangulation (M-triplet features) namely M3gl (Miguel et al 2012). The findings from this research can be used to provide guidance in the design, development, implementation and evaluation of an open, flexible finger print matching algorithm for biometric identification in Kenya.

The development of the solution followed standard software development model as shown in the Figure 14 below;

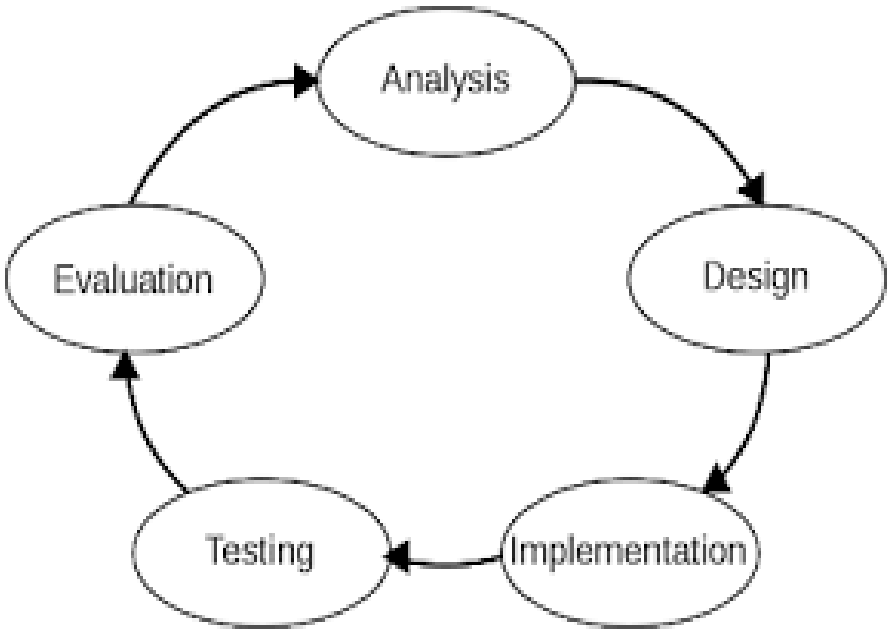


Figure 14 –Software development process flow

3.2 Requirements Analysis

Requirements analysis was done to define what and how the desired system for biometric recognition would be designed to address the research objectives. A review of existing documentations on biometric recognition systems was done. This helped in identifying gaps in the requirements and in the design of the proposed solution. Some of the systems documentations that were studies included the Kenya Biometric Voter registration system, SourceAFIS system and the National Institute of Science and Technology (NIST) algorithms. Figure 15 below describes the process followed in the requirements analysis.

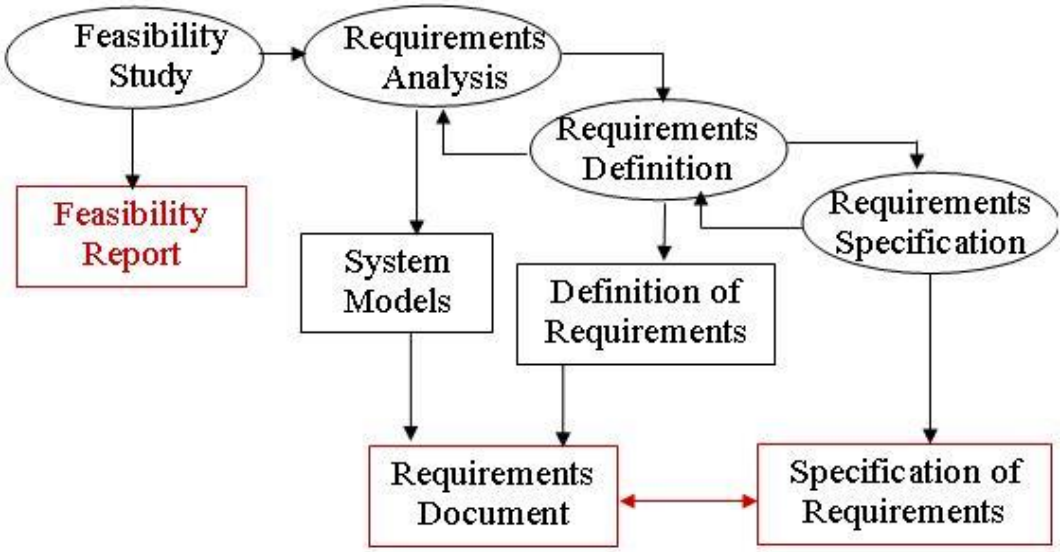


Figure 15- Requirements Analysis Process

During this stage, the following requirements were identified;

3.2.1 Functional Requirements

These defined how the system should function from the end-user's perspective. They described the features and functions with which the end-user will interact with the system directly. It included the following;

i. Provide an interface with menus for selection of different functional operations like file menu, window and help menus.
ii. Provide and interface for conversion of WSQ images into BMP images. This entails creating a form where the user specifies the directory path for the source images and a directory path for the destination / output images
iii. Provide an interface for displaying fingerprint features for visual comparisons
iv. Provide an interface for automatic matching of fingerprints and displaying similarity scores, time taken and number of matching features.

3.2.2 Non-Functional Requirement

These are those requirements that define the quality characteristics of the system. They specify criteria that judge the operation of a system. They included the following aspects of the system;

i. Response time: The system should be able to perform matching of at least 1000 records within thirty (30) minutes
ii. Capacity : The system should be able to match a large dataset of at least 1000 candidate fingerprints
iii. Usability : The system should be easy to learn and use with minimal user assistance
iv. Scalability: The system should be scalable to perform matching of unlimited number of finger print records without having to change the code or re-program
v. Data Integrity: The system should ensure that the original fingerprint images are maintained i.e. the system should not modify or tamper with the original fingerprint images.

3.3 Interface Design

In order to develop the required solution, an interface design of the prototype was developed with the various screens required to achieve the required functions. A screen for the main menu was drawn as shown in the Figure 16 below.

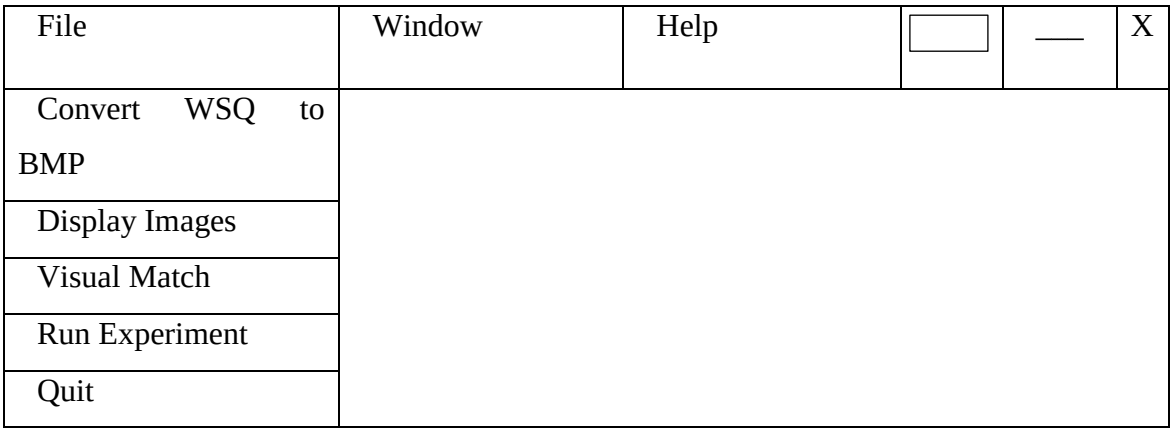


Figure 16 - Main Form

3.3.1 Main form

The main form (figure 17) provides menus form accessing various functions within the application e.g. file menu for performing system functions, window menu for arraigning multiple windows and help to the user assistance and help functions.

The figure 17 below represents the main menu

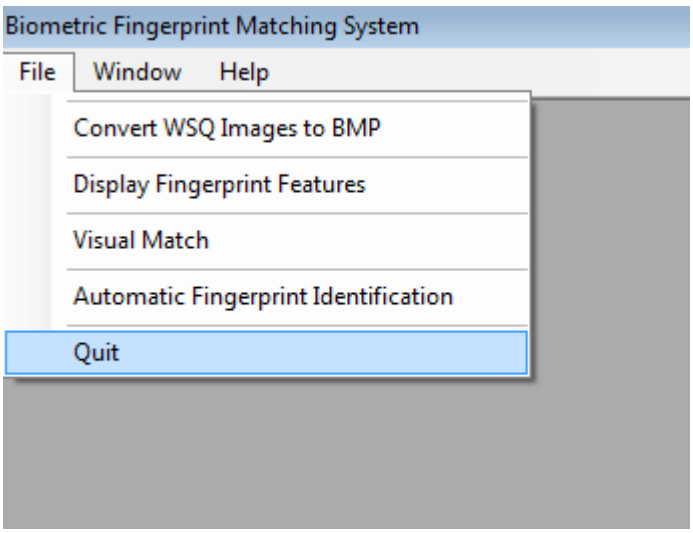


Figure 17 - Main Menu

3.3.2 Matching Experiment Form

The experimental matching form provides the interface to select the required algorithm for the matching test and execute the match. The matching process runs in the background and displays the results on the screen in a grid format. The results of the matching experiment are then saved into comma-separated values (CSV) file in a windows directory. The Figure 18 below shows the form for executing the experiment.

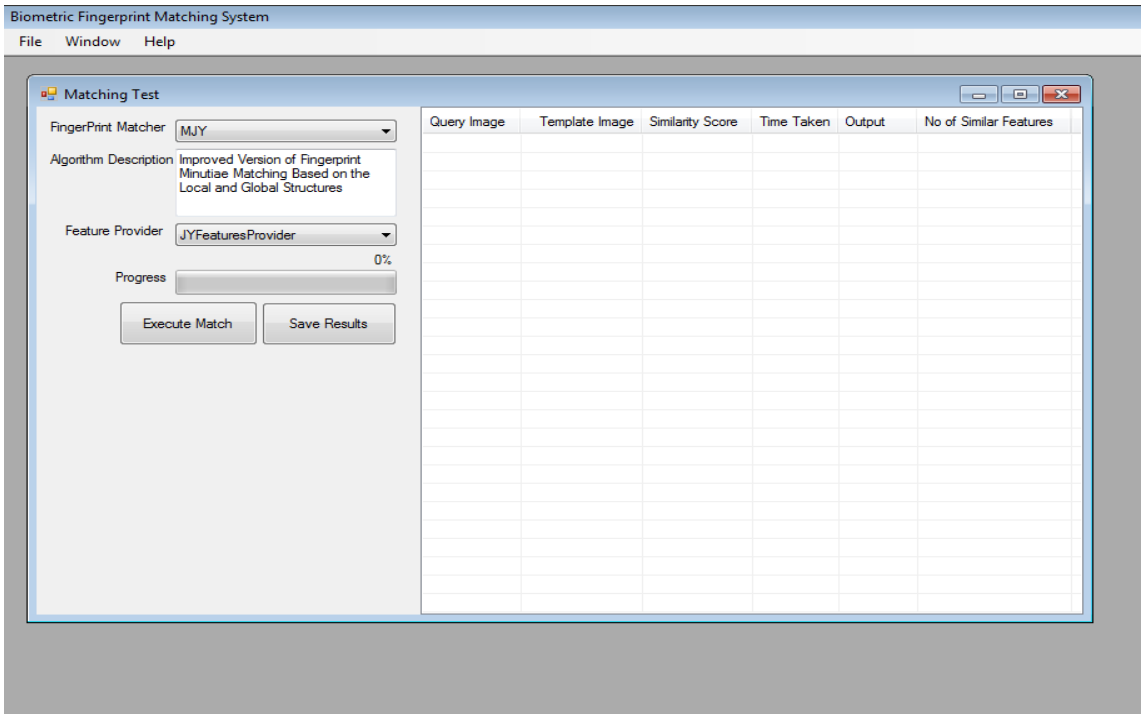


Figure 18 - Experiment Form

The figure 19 below shows how the results of the experiment are then saved

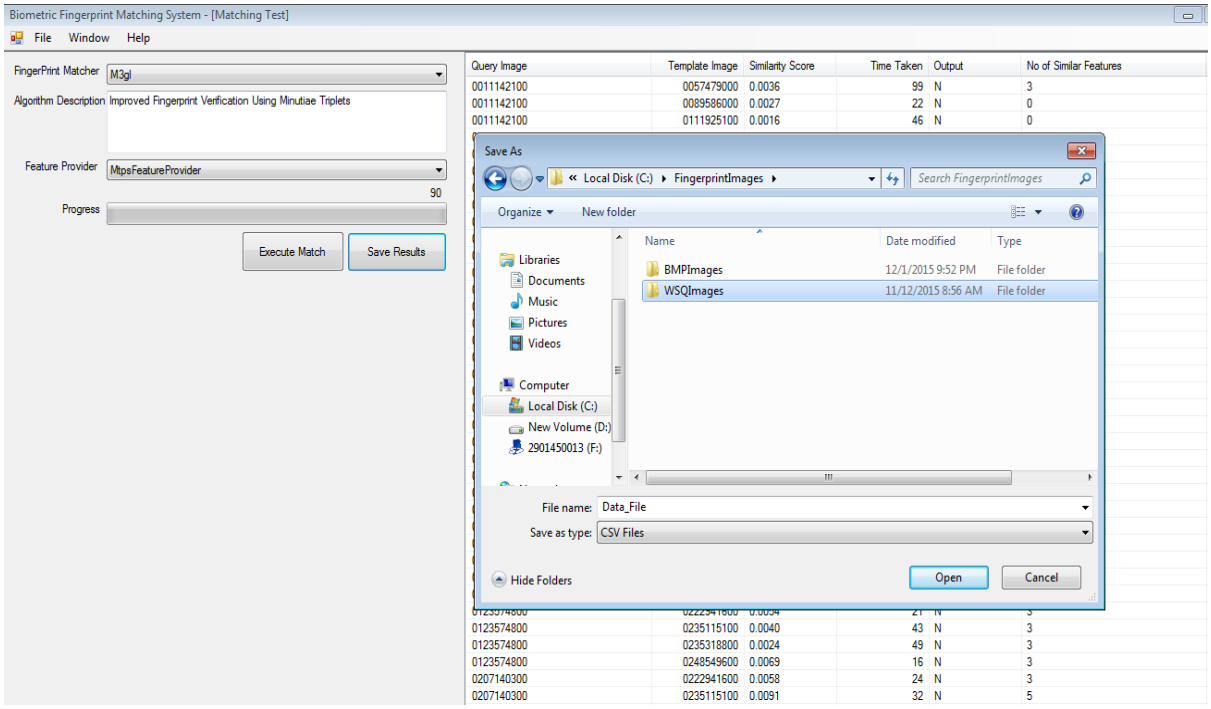


Figure 19 - Save Results Dialog Box

3.4 Data Collection and Conversion

The data used for this research was based on a collection of image impressions of the four (4) fingers from each candidate individuals. The data is captured using a 500 dpi resolution finger print scanner (4-4-2) that produced images of 512x480 pixels. Prior to the features extraction and matching process, each image is passed through a pre-processing stage to improve on the image quality. For ease of identification and matching, each image is assigned a unique name as follows;

i. Left Thumb: xxxxxxxxxx_31.wsq
ii. Left Index: xxxxxxxxxx_32.wsq
iii. Right Thumb: xxxxxxxxxx_36.wsq
iv. Right Index: xxxxxxxxxx_37.wsq

The .wsq fingerprint images of 100 candidates are sampled and save in a windows folder in C:\FingerprintImages\WSQImages, A program is developed in c# to convert the .wsq images to bitmap images for the algorithms to read and extract features. The converted bitmap images are stored under C:\FingerprintImages\BMPImages.

The figure 20 below illustrates an interface for the conversion of images from WSQ to Bitmap images that can be processed by the algorithms.

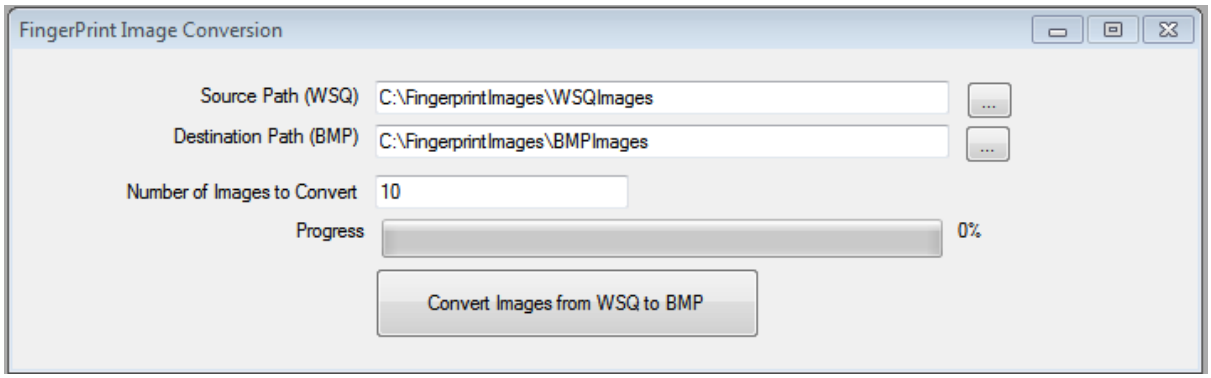


Figure 20 -Image conversion screen

3.5 Sampling Technique

The identification of a person requires the comparison of his/her fingerprint with all the fingerprints in a database, which may be very large (several million fingerprints). A common strategy to reduce the number of comparisons during fingerprint retrieval and to improve the response time of the identification process is to divide the fingerprints into some predefined classes.

In this research, we randomly selected a sample of four (4) fingerprints (Left index, Left Thumb, Right Index, Right Thumb) from amongst 100 randomly selected candidates in the database of registered individuals. The assumptions and considerations in this research were that all the fingerprint capture equipment have the same configurations for the registration software, secondly, it is assumed that the environmental conditions did not affect the quality of the captured fingerprints and therefore the conclusions drawn from the samples were generalized into the entire population.

3.6 Algorithm Implementation

The algorithms were implemented on a Microsoft visual studio 2010 development environment and coded as windows forms application interface. Various pre-programed standard libraries were referenced and used for basic windows form functions which included open dialog and save dialogue boxes. Other fingerprint feature extraction routines and image processing libraries were also extensively used. Some of the libraries included the windows form dialogues, text manipulation functions and input/output libraries. Special libraries included the wsq2bmp decoder libraries and Rather1995 image processing routines.

3.7 Prototype Testing

Each candidate fingerprint was compared with every member in the population set without repetition. The process was automated by ensuring that each query fingerprint is selected from the population and matched against each and every finger print in the entire population. This process was able to uniquely discriminate fingerprints belonging to the same individual and thus identifying duplicate registrations either as fraudulent or erroneous enrolments. In this research, the experiment was conducted by executing each algorithm against a database of sampled fingerprints and obtaining values of similarity scores, time taken and the number of similar features obtained from the query and template fingerprints. The outputs were then analysed using SPSS statistical tool to obtain statistical values for comparison. Fingerprint features were extracted from each of the four sampled fingerprints (Left Thumb, Left Index, Right Thumb, Right Index) and average similarity scores were computed.

For each Candidate Q[i] , i=0…….N-1 where N is the total number of candidates, the finger prints features from each of the four (4) fingers of a candidate were extracted and compared with the template fingerprints from the entire set of candidates Q[i], where i=1 …..N, in this case N was 40.

Figure 21 and 22 shows the left hand fingerprints with numbering of the fingerprints

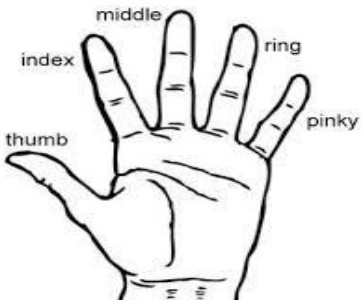


Figure 22 - Left Hand

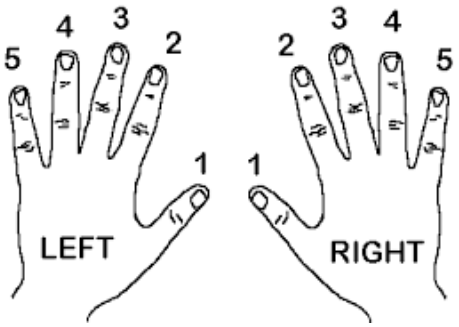


Figure 21 -Labelling of the Fingers

The experiment was conducted by running the routine that picks one fingerprint from the dataset at a time and matching it against each and every other fingerprint as illustrated in table 1 and table 2 below.

Table 1 - Query Fingerprint

QUERY FINGERPRINT (Index)				
Left Thumb	x =0	x+1	… …	n-1
Left Index	x =0	x+1	… …	n-1
Right Thumb	x =0	x+1	… …	n-1
Right Index	x =0	x+1	… …	n-1

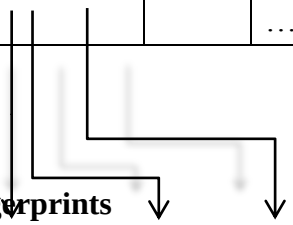


Table 2 - Template fingerprints

TEMPLATE FINGERPRINT (Index)				
Left Thumb	i =x+1	i+1	… …	n
Left Index	i =x+1	i+1	… …	n
Right Thumb	i =x+1	i+1	… …	n
Right Index	i =x+1	i+1	… …	n

An analysis of the order of computations was done as shown in equation (9) below. It was observed that the number of comparisons performed by the matching algorithm was in the order

$$(n-1) + (n-2) + (n-3)\ldots+1, = n*(n-1)/2. \quad \ldots\ldots (9)$$

For example, given a dataset of 5 fingerprints, the number of comparisons done would be calculated as follows

$$5*(5-1)/2 = 5*4/2=5*2=10 \text{ Comparisons.} \quad \ldots\ldots (10)$$

3.8 Visual Match Test

Visual matching involved selecting a query fingerprint (Q) and a template Fingerprint (T) then executing a match test. The results indicated accuracy between 0 (Lowest similarity) and 1(Highest similarity). Using each of the algorithms implemented, the algorithms are able to uniquely match two fingerprints by calculating the accuracy of match between them as well as displaying the number of similar features. When two fingerprints are identical, the

similarity score will be equal to 1 and the number of similar minutiae will be a value above 20 as shown in Figure 23 below.

We were therefore able to visually determine the degree to which the algorithm is able to discriminate one fingerprint from another.

Figure 23 -Visual match of the same fingerprints

When the Query fingerprint and the Template fingerprint are not from the same finger or individual, the similarity score is below the 0.2 and the number of matching minutiae is low (below 10) as shown in the figure 24 below.

Figure 24-Visual Match of different fingerprint images

CHAPTER FOUR – RESULTS AND DISCUSSIONS

4.1 Introduction

This chapter describes the findings and results of the research within the scope of the research objective. The results in this chapter show the performance indicators in terms of algorithm speed and accuracy. This helps to clarify the implications on performance of the fingerprint matching implemented in solution.

4.2 Overall Results

After the tests were successful, the image conversion program was executed to convert the four (4) fingerprint images for 100 candidates. This process took approximately two (2) minutes. The main experimental program for the matching and de-duplication was performed where the de-duplication using MWQY algorithm took about 45 minutes and the M3gl algorithm took an approximate 3 minutes to complete. The Output value indicated whether there was a positive (P) Match or Negative (N) Non-match result /Impostor

An extract of the sample results are given in the Table 3.

Table 3- Sample Matching Results using Minutia Triangulation

Query Image	Template Image	Accuracy (Similarity Score)	Time Taken (Milliseconds)	No of Features	Output
11142100	57479000	0.0036	80	3	N
11142100	89586000	0.0027	22	0	N
11142100	111925100	0.0016	25	0	N
11142100	123574800	0.0028	14	3	N
11142100	207140300	0.0014	13	0	N
11142100	222941600	0.0059	23	3	N
11142100	235115100	0.0023	26	0	N
11142100	235318800	0.0038	30	4	N

An extract of the sampled results of the experiment using minutia and global orientation based algorithms is shown in the Table 4 below

Table 4 - Sample Matching Result using Minutia and Global Orientation Feature

Query Image	Template Image	Accuracy (Similarity Score)	Time Taken	No of Features	Output
11142100	57479000	0.1329	703	3	N
11142100	89586000	0.1313	710	6	N
11142100	111925100	0.1232	856	6	N
11142100	123574800	0.0975	543	1	N
11142100	207140300	0.1146	481	1	N
11142100	222941600	0.155	704	4	N
11142100	235115100	0.1194	747	4	N
11142100	235318800	0.0933	125	4	N
11142100	248549600	0.2047	511	4	N

4.3 Analysis of Results Using Frequency Distribution Graphs

The simulation of matching experiment was conducted for each algorithm using a dataset of 100 sampled candidates. Frequency distribution graphs in below were created that shows the distribution of the three parameters accuracy, time taken and number of similar features. The Accuracy values range between zero and one (0…1) with 0 being no similarity between the query and template image while 1 indicates high similarity.

4.3.1 Graphs for Minutia Triangulation Algorithm (M3gl)

The Figure 24 below represents the graph of accuracy degrees ranging between 0 – 1 on the x-axis and the frequencies of occurrence on the y-axis where accuracy was measured in terms of similarity scores.

From the figure 25 below, it can be observed that the minimum value is 0.0009 and the maximum value is 0.0142 with the majority values ranging between 0.0021 and 0.0065.

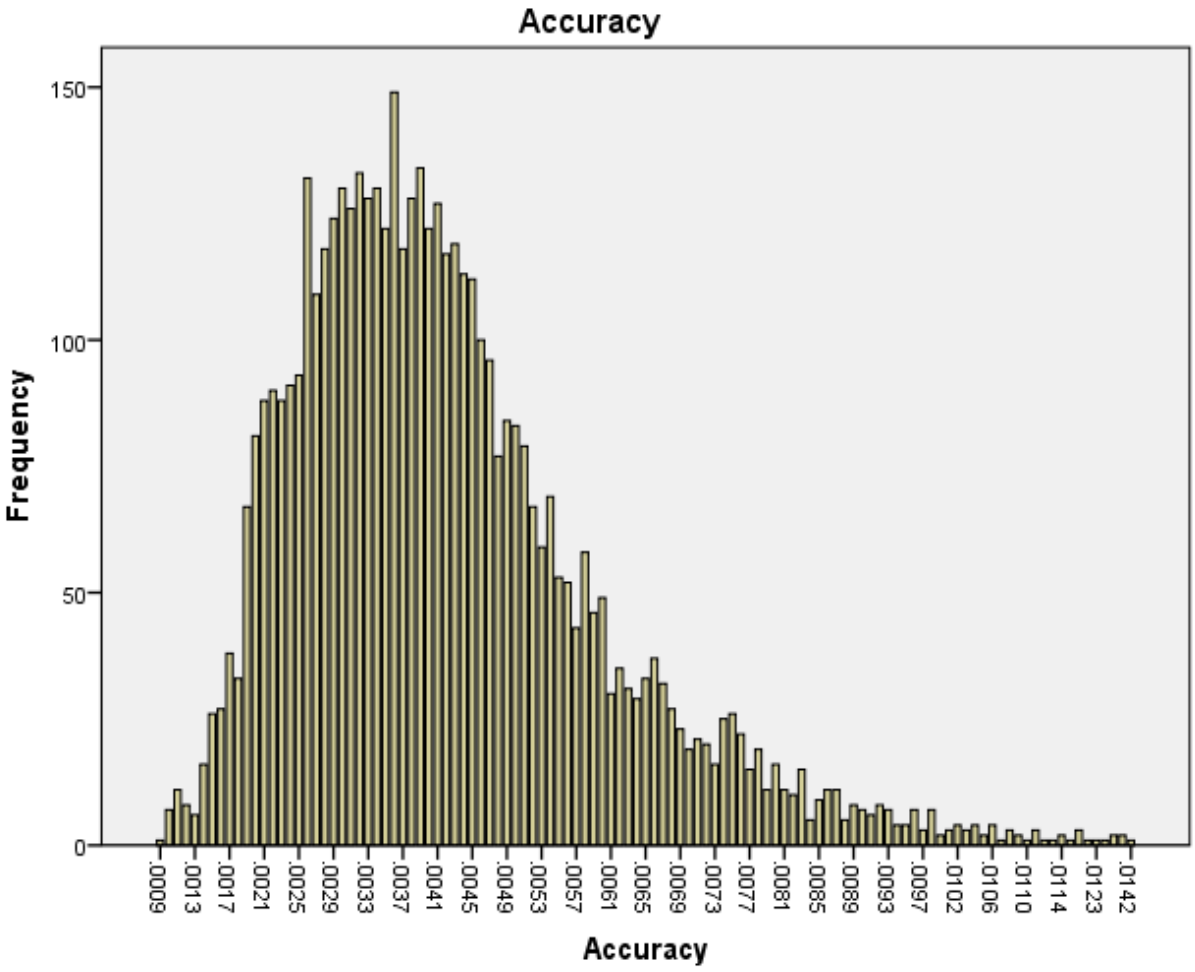


Figure 25-Minutia Triangulation - Bar Chart on Accuracy

The Figure 26 below represents the graph of Time taken to perform a match with the M3gl algorithm. The values ranges between 0 – 400 milliseconds on the x-axis and the frequencies of occurrence on the y-axis. It was noted the graph is skewed to the left with majority of the frequencies falling between 10 and 70 milliseconds

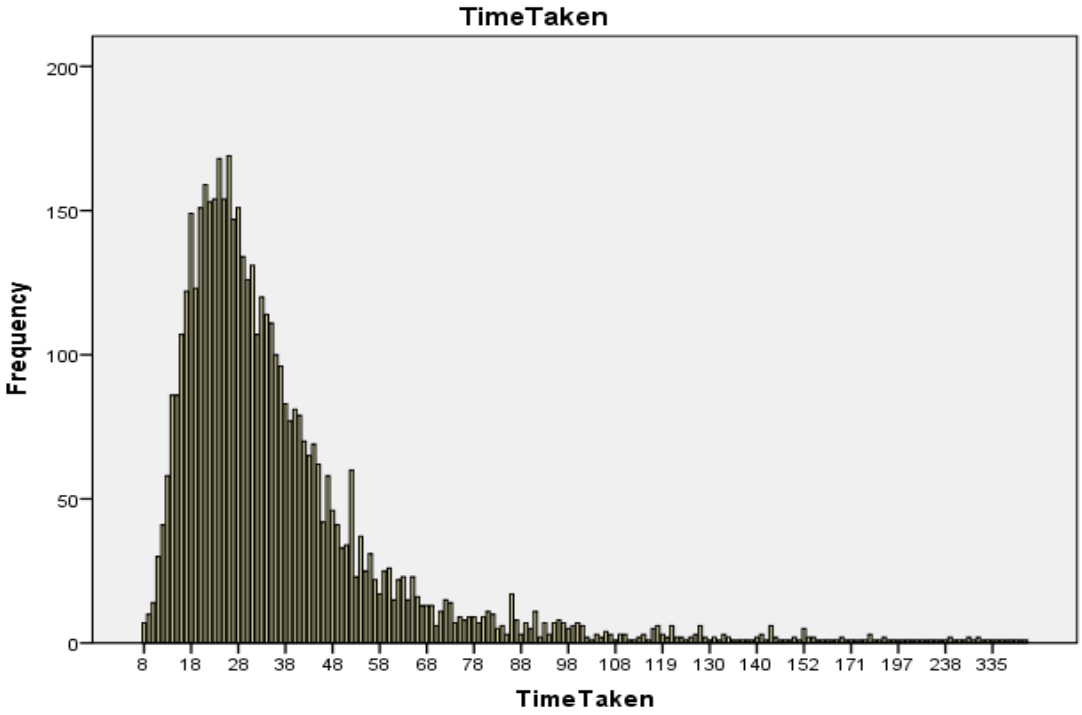


Figure 26- Minutia Triangulation - Bar Chart on Time taken (ms)

The figure 27 below show the distribution graph of the number of similar features identified by the algorithm on the x-axis against the frequency of occurrence on the y-axis. It can be noted that the features were fewer in this algorithm.

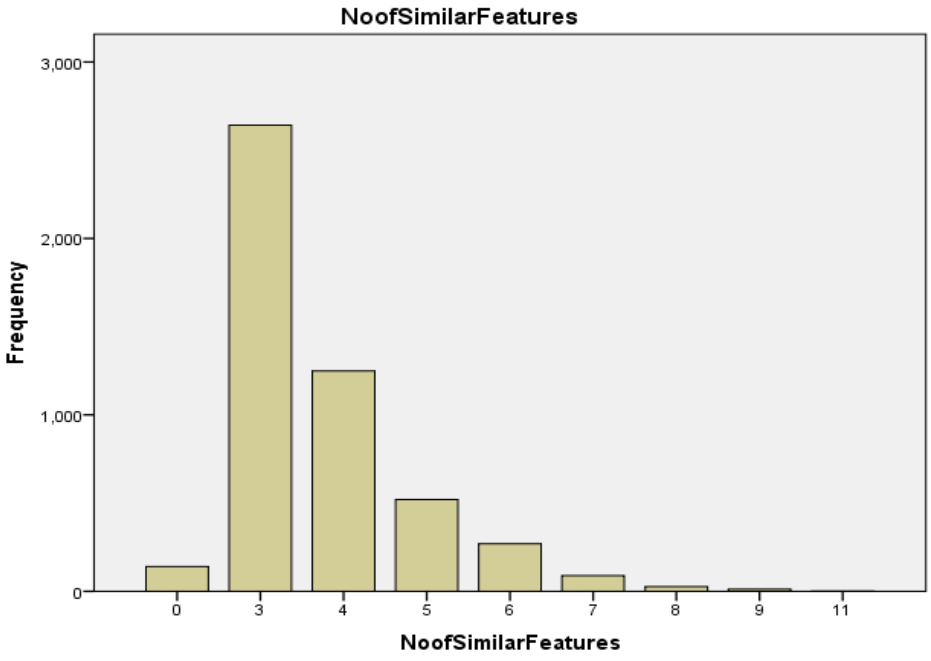


Figure 27 -Minutia Triangulation - Bar Chart on No of similar features

4.3.2 Graphs for Minutia and Global Orientation Algorithm

A second experiment that was conducted using the minutia and Global orientation based algorithm named MQYW, similar graphs were produced. In figure 28 below, the graph represents accuracy of comparison versus frequency. It can be observed that average accuracy falls in the range 0.10 to 0.18 indicating a higher degree accuracy as compared to the previous algorithm.

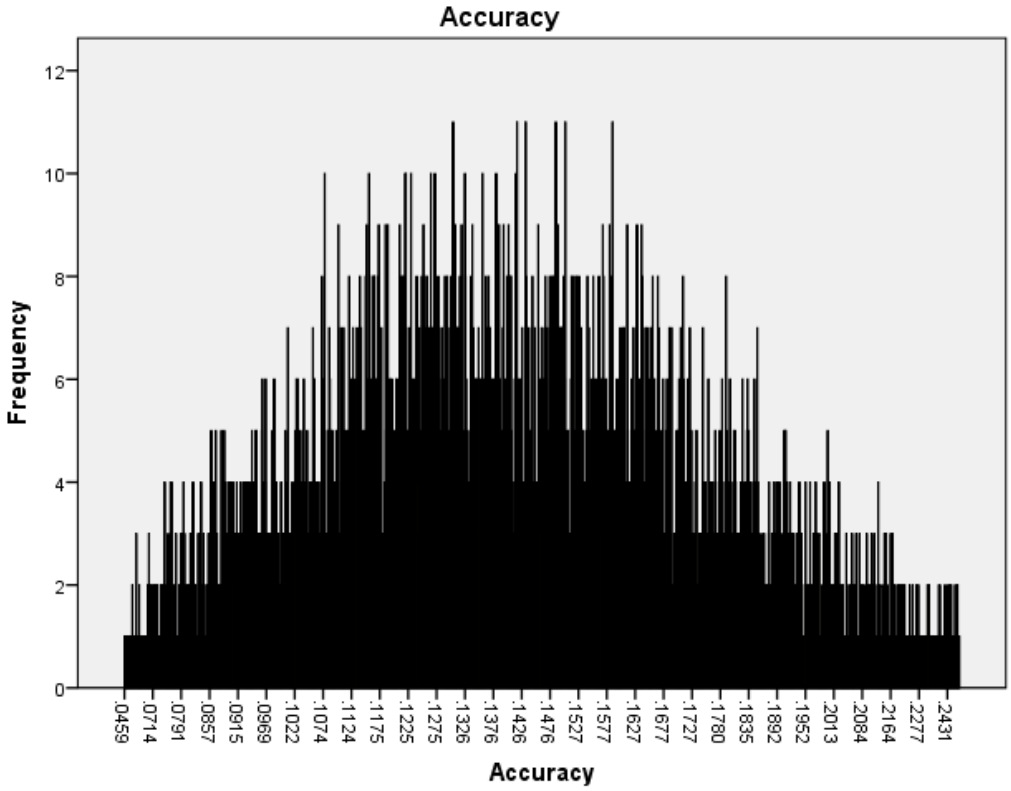


Figure 28 -Global Orientation - Bar Chart on Accuracy

In terms of time taken to perform comparisons, the graph in figure 29 below indicates that this algorithm performs poorly. Majority of the comparisons are performed in the range between 439 milliseconds and 890 milliseconds.

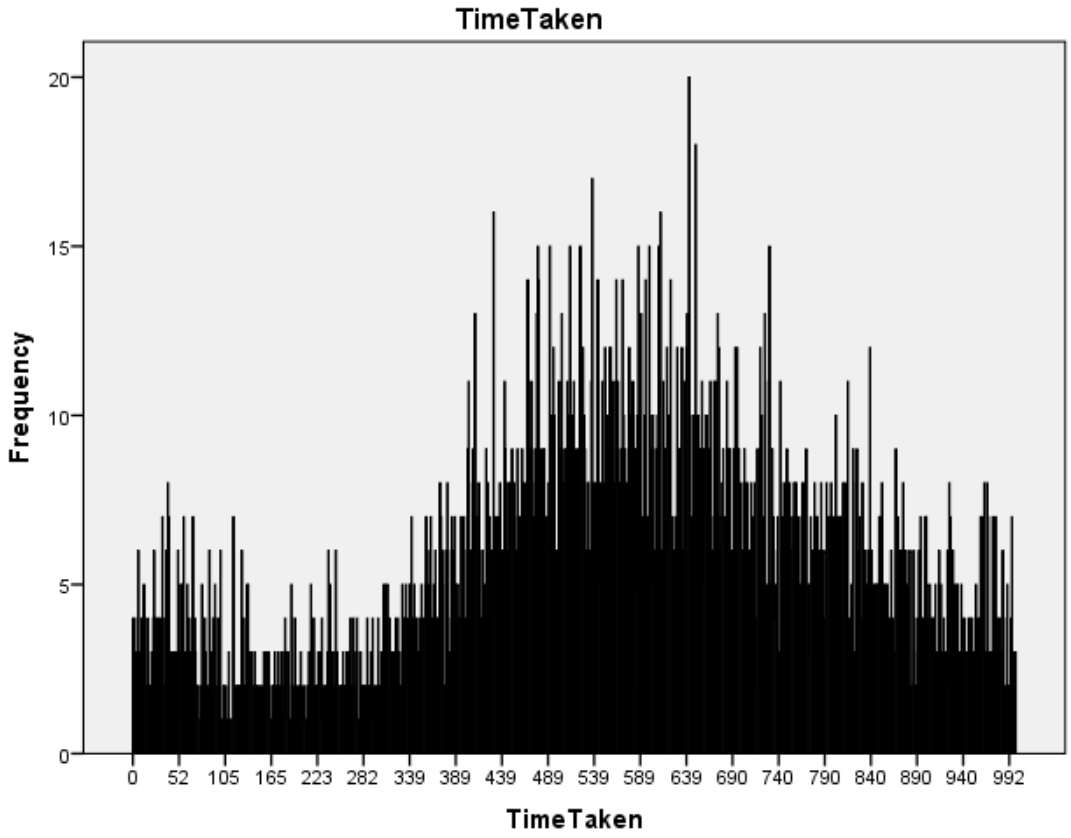


Figure 29 -Global Orientation - Bar Chart on Time taken (ms)

The figure 30 below compares shows number of similar features identified against the frequency. This algorithm is able to discriminate and identify more features than the previous algorithm indicating a higher degree of accuracy.

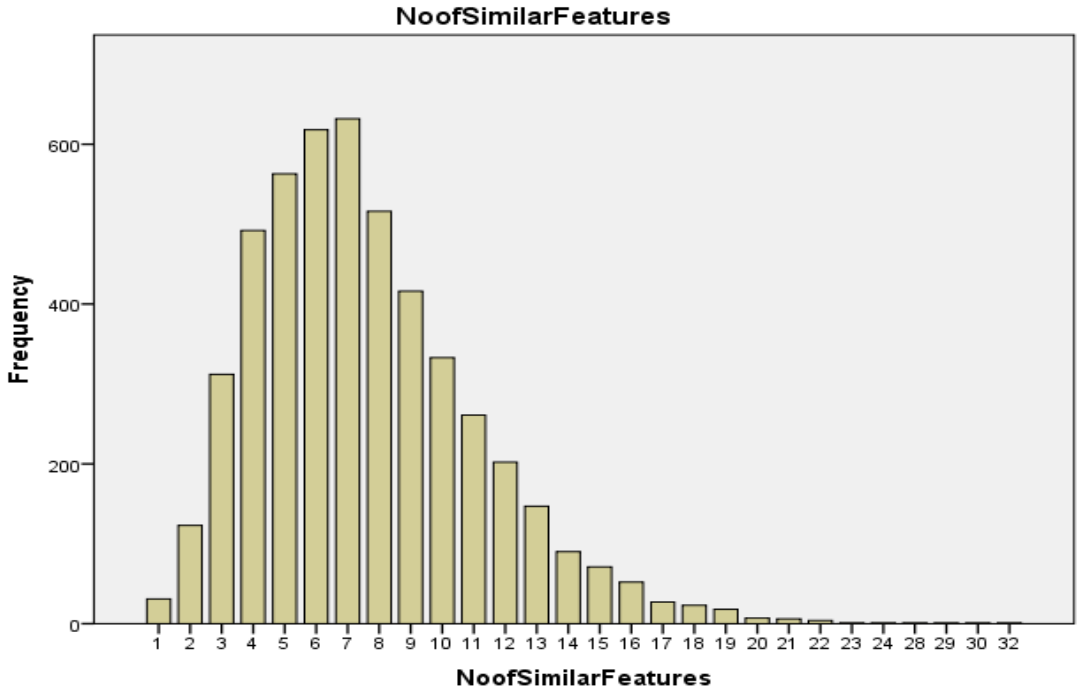


Figure 30 -Global Orientation - Bar Chat on No of Similar Features

4.4 Discussions

Using the descriptive values of minimum, maximum, average and the standard deviation of the three variables we were able to compare and analyse the performance of the two algorithms as shown in table 5 below. The table 5 and table 6 below shows that a total of 4950 comparisons were performed and values for accuracy, time taken and number of similar features identifies were computed for each algorithm

Table 5 - Minutia Triangulation (M-Triplet) Based

M3gl Algorithm - Descriptive Statistics

	N	Minimum	Maximum	Mean	Std. Deviation	Variance
Accuracy	4950	.0009	.0142	.004202	.0017840	.000
TimeTaken (ms)	4950	8	504	38.32	31.834	1013.416
No. of Similar Features	4950	0	11	3.66	1.229	1.511
Valid N (Listwise)	4950					

Table 6 -Minutia and Global Orientation Based

MQYW Algorithm - Descriptive Statistics

	N	Minimum	Maximum	Mean	Std. Deviation	Variance
Accuracy	4950	.0459	.3009	.142433	.0363188	.001
TimeTaken	4950	0	999	563.76	237.459	56386.784
NoofSimilarFeatures	4950	1	32	7.56	3.532	12.475
Valid N (listwise)	4950					

Where N= Number of Fingerprints comparisons performed in a dataset of 100 candidates with four fingerprints each.

This research has revealed that fingerprint matching algorithms require high computational resources in terms of speed, time and memory. For instance, it took approx. 30mins to perform de-duplication of one hundred (100) candidates with four (4) fingerprint images each on a personal laptop. This research was successfully implemented on a personal laptop computer based on simulation of algorithms implemented on C# using Microsoft Visual Studio 2010 professional development environment. It is observed that fingerprint matching based on minutia triangulation algorithms performs better in terms of speed with an average of **38.32** milliseconds as compared to matching based on a combination of minutia and global orientation features with an average of **563.76** milliseconds.

In terms of accuracy of matching, the algorithms based on a combination of minutia and global orientation field features performs better with an average similarity score of **0.142433** as compared to m-triplet based matching with an average similarity score of **0.004202**

CHAPTER FIVE - CONCLUSIONS AND RECOMMENDATIONS

5.1 Introduction

This section summarizes the achievements, recommendations and future work in this area. Specifically, the conclusions drawn from these results can be used in the design of future biometric recognition system in citizen identification, paternity tests, criminal investigation as well and access control systems.

5.2 Achievements

In this research, we were able to analyse and effectively compare the performance of two fingerprint matching algorithms a) one that is based on minutia features and b) another based on minutia triangulation features. Both of these algorithms have been compared in terms of speed of comparison and degree of accuracy.

In this research, we also developed and implemented a prototype for fingerprint matching to identify duplicate records using four (4) fingerprint images from 100 candidates. The experiment was executed in a controlled environment using some free libraries available on the internet. The prototype developed was able to extract and convert fingerprint images captured using standard enrolment devices with 500 DPI.

The research has helped re-affirm the fact that the choice of the fingerprint matching algorithm certainly improves the matching performance of the fingerprint based recognition system. Other factors affecting the performance of fingerprint recognition systems have widely been documented namely the quality of fingerprint images, enhancement techniques applied. A lot of research efforts has previous been focused towards enhancing the quality of the fingerprint images, improving the enrolment process and enhancing the usability of the recognition software.

From this research, it has been revealed that the choice of an appropriate algorithm is largely dependent on the intended application of the biometric recognition system, the resources available as well as the desired level of accuracy. The computational resources available for carrying out large scale biometric matching affects in a large way the choice of the matching algorithm.

5.3 Recommendations

5.3.1 Recommendations for Research

We recommend that further comparative analysis of neural network and correlation based algorithms be undertaken using the same parameters to identify their performance as compared to minutia based fingerprint matching techniques. On a similar note, further comparative analysis needs to be done using a combination of accuracy, speed, and memory requirements in order to determine the correlation between speed and memory requirements of a matching algorithm. Further comparative analysis on the computational efficiency using ROC curves and confusion matrix should also be done to advance knowledge on this area.

5.3.2 Recommendations for Practice

Biometric recognition is a new field that is gaining popularity in the Kenyan market. Every industry in Kenya is adopting biometric techniques in identifying both their staff and their clients. It is in this perspective that prudent practices and techniques should be adopted in the selection of solutions for biometric recognition. From the findings of this research, we can recommended institutions and organizations that intend to acquire Biometric Recognition systems should consider evaluating the solution proposals from vendors based on the degree of accuracy, speed of matching as well as the memory space requirements. These factors will to a large extent depend on the estimated population size in terms of the number of fingerprint records to be processed at any one time as well as the processing resources available.

5.4 Future Work

This project provides introductory concepts to fingerprint recognition and matching based on minutiae point matching. The tested conducted in this research can be improved to ensure their applicability in an enterprise scale. For instance, the interface can be enhanced to provide functionality performing matching using the WSQ images without having to convert them to bitmap images. This will require purchase of the WSQ image library available from Cognaxon and integrating the same into the application. The other improvements that can be done include; a) Implementation of parallel processing using threading in the matching algorithm; b) Improvement of the minutiae templates by applying other image enhancement routines and c) Using fingerprint classification to speed up the algorithm.

5.5 Limitations of this Research

In this Research, we did not use memory requirements as a factor in the evaluation of the performance of an algorithm. It is also important to note that in this research, we did not analyze the performance of the algorithms using the confusion matrix that is based on True Match and False Match rate due to time constraints.

Most of the fingerprint de-duplication systems are based on proprietary source codes and thus obtaining these source codes for bench mark analysis to determine their architecture design was not possible while some were available at a fee. However, there are a number of projects that provide free and open source fingerprint recognition tool kits that can be used to build de-duplication prototypes e.g. www.sourceAfis.org.

REFERENCES

Chaohong W (2007), Advanced feature extraction algorithms for automatic fingerprint recognition systems. *A dissertation submitted to the faculty of the graduate school of state university of New York at buffalo in partial fulfilment of the requirements for the degree of doctor of philosophy*.

Delac K & Grgic, M (2004) "A survey of Biometric recognition methods", *Electronics in marine.* 46th International Symposium, vol. no. pp.184,193.

Pato J N & Millettee L. I (2010), Whither Biometrics Committee; National Research Council, *Biometric recognition – challenges and opportunities.*

Jain K., Hong L., Pankanti S., Bolle R., (1997), An identity – authentication system using fingerprints, Proc. IEEE 85 (9) ,pp 1364-1388,

Qi J., Yang S & Wang Y (2005), Fingerprint matching combining the global orientation field with minutia," Pattern Recognition Letters, vol. 26, pp. 2424-2430,

Kumar, D. A. & Begum, T. U. S. (2013). A Comparative Study on Fingerprint Matching Algorithms for EVM. Journal of Computer Sciences and Applications, 1(4), 55-60.

Kenneth K, Stanley J, Michael M, Charles L., Elham T, Michael D., Craig I., (2007), *User's Guide to NIST Biometric Image Software*, NIST Interagency/Internal Report (NISTIR) – 7392.

Lukas W. (2009), A minutiae-based matching algorithm in fingerprint recognition systems, journal of medical informatics & technologies vol. 13/2009, issn 1642-6037

Medina P, Garcia, M., et.al, (2011), Robust fingerprint verification using m-triplets, International Conference on Hand-Based Biometrics, Hong Kong, 2011, pp. 1-5.

Nadaraja M, Celalettin T, et. al, (2011), Fingerprint Biometric for Identity management International Journal of Industrial Engineering and Management (IJIEM), Vol. 2 No 2, 2011, pp. 39-44 < *http://www.ftn.uns.ac.rs/ijiem/*>

Germain R. S., Califano A., & Colville S. (1997) Fingerprint matching using trans-formation parameter clustering. *Computational Science and Engineering, IEEE Computing in Science & Engineering*, 4(4):42–49,

Roli B, Priti S, & Punam B. (2011), Minutiae Extraction from Finger print Images – A review, International Journal of Computer science , vol. 8 Issue 5, No 3. Available from: *www.ijcsi.org/papers/IJCSI-8-5-3-74-85.pdf:*last accessed 12 March 2015

Saleh A.A., Adhami R.R., (2001), Curvature-based matching approach for automatic fingerprint identification, Proceedings of the Southeastern Symposium on System Theory, pp. 171-175,.

X. Jiang, W.-Y. Yau, and W. Ser. (2001), Detecting the fingerprint minutiae by adaptive tracing the gray-level ridge. *Pattern Recognition*, 34(5):999–1013,

X. Jiang & W. Y. Yau, (2000), Fingerprint Minutiae Matching Based on the Local and Global Structures, in 15th International Conference on Pattern Recognition, Barcelona, Spain, pp. 1038-1041.

APPENDIX I– Sample Output for M3gl Algorithm

Sample Output for M3gl Algorithm					
Query Image	Template Image	Similarity Score	Time Taken	Output	No of Similar Features
11142100	57479000	0.0036	80	N	3
11142100	89586000	0.0027	22	N	0
11142100	111925100	0.0016	25	N	0
11142100	123574800	0.0028	14	N	3
11142100	207140300	0.0014	13	N	0
11142100	222941600	0.0059	23	N	3
11142100	235115100	0.0023	26	N	0
11142100	235318800	0.0038	30	N	4
11142100	248549600	0.0032	16	N	0
11142100	254476300	0.0035	23	N	3
11142100	257861600	0.0029	59	N	4
11142100	258269000	0.0028	28	N	3
11142100	258734600	0.0074	20	N	3
11142100	263293600	0.004	16	N	3
11142100	267619800	0.0111	21	N	7
11142100	273915100	0.0047	29	N	3
11142100	276301300	0.0069	16	N	3
11142100	289435100	0.004	17	N	3
11142100	297020500	0.0027	19	N	0
11142100	298853800	0.0076	22	N	3
11142100	311046700	0.0035	31	N	3
11142100	338924500	0.0042	16	N	3
11142100	360148100	0.0066	16	N	4
11142100	360536100	0.006	23	N	5
11142100	378442300	0.0043	18	N	3
11142100	392769200	0.0039	17	N	3
11142100	394660700	0.0038	19	N	0
11142100	402527400	0.0021	24	N	0
11142100	421413300	0.0029	22	N	3
11142100	440687200	0.0051	14	N	0
11142100	465218500	0.0042	38	N	3
11142100	473647800	0.0066	23	N	3
11142100	487974700	0.0046	22	N	3
11142100	493823800	0.003	25	N	3
11142100	508742400	0.0056	22	N	4
11142100	512680600	0.0016	17	N	0
11142100	525581600	0.0101	15	N	6
11142100	529490700	0.0047	27	N	3
11142100	532264900	0.0024	23	N	3
11142100	550316600	0.0043	22	N	4

Sample Output for M3gl Algorithm					
Query Image	Template Image	Similarity Score	Time Taken	Output	No of Similar Features
11142100	552508800	0.0016	34	N	0
11142100	559172700	0.0039	25	N	3
11142100	575177700	0.0044	16	N	3
11142100	583044400	0.0038	18	N	4
11142100	600485000	0.002	24	N	0
11142100	616926500	0.0054	31	N	3
11142100	622630100	0.005	21	N	3
11142100	637636000	0.004	23	N	0
11142100	638402300	0.003	16	N	3
11142100	660101200	0.0034	12	N	0
11142100	681286000	0.0025	21	N	3
11142100	682285100	0.0041	34	N	0
11142100	682925300	0.0016	34	N	0
11142100	703178900	0.0036	20	N	0
11142100	713053500	0.0049	33	N	3
11142100	719445800	0.0038	32	N	3
11142100	729960600	0.0023	21	N	0
11142100	733763000	0.0039	31	N	3
11142100	739621800	0.0033	29	N	3
11142100	750049300	0.0042	33	N	3
11142100	755879000	0.0026	41	N	0
11142100	763192800	0.0032	25	N	3
11142100	766985500	0.0067	18	N	3
11142100	774454500	0.0044	15	N	0
11142100	777674900	0.0039	23	N	3
11142100	800712400	0.0023	21	N	0
11142100	825689900	0.0067	18	N	0
11142100	852937200	0.0046	19	N	3
11142100	869388400	0.0049	9	N	3
11142100	871726100	0.005	15	N	3
11142100	886567100	0.0038	9	N	0
11142100	887682600	0.0014	31	N	0
11142100	890796300	0.0047	16	N	3
11142100	909323300	0.0031	21	N	3
11142100	914735900	0.0035	14	N	0
11142100	935523000	0.0052	20	N	4
11142100	942739800	0.0035	18	N	3
11142100	945029000	0.0034	25	N	3
11142100	977349400	0.006	14	N	3
11142100	990512300	0.0019	15	N	0
11142100	1003423000	0.0021	20	N	0
11142100	1004053500	0.0038	18	N	3
11142100	1035568800	0.0036	14	N	0
11142100	1039875600	0.0073	19	N	4

Sample Output for M3gl Algorithm					
Query Image	Template Image	Similarity Score	Time Taken	Output	No of Similar Features
11142100	1050177000	0.0033	37	N	3
11142100	1062360200	0.005	15	N	3
11142100	1094118000	0.0038	22	N	3
11142100	1118426200	0.0015	56	N	3
11142100	1138980500	0.0028	15	N	0
11142100	1145586200	0.0041	39	N	3
11142100	1153976700	0.003	20	N	0
11142100	1173929600	0.0056	12	N	0
11142100	1183590800	0.0035	21	N	3
11142100	1186403800	0.0045	19	N	3
11142100	1188111000	0.0044	28	N	3
11142100	1206240300	0.003	27	N	3
11142100	1215261300	0.0021	34	N	3
11142100	1226561800	0.0022	11	N	0
11142100	1226668500	0.0013	19	N	0
57479000	89586000	0.004	37	N	3
57479000	111925100	0.0039	41	N	3
57479000	123574800	0.0063	20	N	5
57479000	207140300	0.0042	21	N	3
57479000	222941600	0.0031	33	N	0
57479000	235115100	0.004	44	N	3
57479000	235318800	0.0027	54	N	3
57479000	248549600	0.004	23	N	3
57479000	254476300	0.0072	38	N	5
57479000	257861600	0.0035	55	N	6
57479000	258269000	0.0029	47	N	3
57479000	258734600	0.0042	24	N	3
57479000	263293600	0.0038	24	N	3
57479000	267619800	0.002	29	N	3
57479000	273915100	0.0032	33	N	0
57479000	276301300	0.0074	23	N	3
57479000	289435100	0.0065	25	N	4
57479000	297020500	0.0045	26	N	3
57479000	298853800	0.0042	26	N	3
57479000	311046700	0.0041	51	N	3
57479000	338924500	0.006	23	N	4
57479000	360148100	0.0059	25	N	3
57479000	360536100	0.0047	32	N	5
57479000	378442300	0.0043	28	N	3
57479000	392769200	0.0029	31	N	3
57479000	394660700	0.0064	27	N	4
57479000	402527400	0.0035	36	N	3
57479000	421413300	0.0045	34	N	4
57479000	440687200	0.0074	19	N	3

Sample Output for M3gl Algorithm					
Query Image	Template Image	Similarity Score	Time Taken	Output	No of Similar Features
57479000	465218500	0.0034	27	N	4
57479000	473647800	0.0017	41	N	0
57479000	487974700	0.0038	35	N	3
57479000	493823800	0.0044	48	N	4
57479000	508742400	0.0034	42	N	3
57479000	512680600	0.0044	29	N	3
57479000	525581600	0.005	28	N	3
57479000	529490700	0.0034	44	N	3
57479000	532264900	0.0041	33	N	5
57479000	550316600	0.0045	38	N	3
57479000	552508800	0.0018	60	N	3
57479000	559172700	0.0031	35	N	4
57479000	575177700	0.005	19	N	0
57479000	583044400	0.0041	24	N	0
57479000	600485000	0.0066	38	N	4
57479000	616926500	0.0016	46	N	3
57479000	622630100	0.0038	25	N	3
57479000	637636000	0.0033	33	N	3
57479000	638402300	0.0034	22	N	3
57479000	660101200	0.0055	18	N	4
57479000	681286000	0.0031	30	N	3
57479000	682285100	0.003	31	N	3
57479000	682925300	0.0027	66	N	3
57479000	703178900	0.0043	31	N	3
57479000	713053500	0.0053	46	N	6
57479000	719445800	0.0055	50	N	3
57479000	729960600	0.0062	29	N	3
57479000	733763000	0.002	44	N	3
57479000	739621800	0.0056	43	N	3
57479000	750049300	0.004	24	N	5
57479000	755879000	0.0076	29	N	3
57479000	763192800	0.0022	43	N	3
57479000	766985500	0.0035	26	N	3
57479000	774454500	0.0043	26	N	3
57479000	777674900	0.0036	34	N	3
57479000	800712400	0.0036	33	N	3
57479000	825689900	0.0048	28	N	5
57479000	852937200	0.0044	29	N	3
57479000	869388400	0.008	17	N	8
57479000	871726100	0.0077	20	N	3
57479000	886567100	0.0033	15	N	3
57479000	887682600	0.0022	42	N	3
57479000	890796300	0.0047	28	N	4
57479000	909323300	0.0023	40	N	3

Sample Output for M3gl Algorithm					
Query Image	Template Image	Similarity Score	Time Taken	Output	No of Similar Features
57479000	914735900	0.0029	18	N	0
57479000	935523000	0.0087	32	N	7
57479000	942739800	0.0045	33	N	3
57479000	945029000	0.004	39	N	3
57479000	977349400	0.006	17	N	3
57479000	990512300	0.0067	19	N	3
57479000	1003423000	0.0043	29	N	4
57479000	1004053500	0.008	26	N	5
57479000	1035568800	0.0053	23	N	4
57479000	1039875600	0.0048	29	N	4
57479000	1050177000	0.0028	55	N	4
57479000	1062360200	0.0033	20	N	0
57479000	1094118000	0.0026	34	N	3
57479000	1118426200	0.001	118	N	3
57479000	1138980500	0.0051	23	N	3
57479000	1145586200	0.0037	64	N	5
57479000	1153976700	0.0027	30	N	3
57479000	1173929600	0.0042	17	N	4
57479000	1183590800	0.0057	33	N	3
57479000	1186403800	0.0048	28	N	0
57479000	1188111000	0.0021	42	N	3
57479000	1206240300	0.0033	39	N	3
57479000	1215261300	0.0031	47	N	4
57479000	1226561800	0.0051	13	N	3
57479000	1226668500	0.0044	34	N	3
89586000	111925100	0.0021	51	N	4
89586000	123574800	0.004	28	N	3
89586000	207140300	0.0028	31	N	4
89586000	222941600	0.0032	32	N	5
89586000	235115100	0.0029	62	N	3
89586000	235318800	0.0024	72	N	4
89586000	248549600	0.0021	23	N	3
89586000	254476300	0.0019	43	N	3
89586000	257861600	0.0023	122	N	4
89586000	258269000	0.0019	66	N	4
89586000	258734600	0.0033	31	N	3
89586000	263293600	0.0033	34	N	4
89586000	267619800	0.0045	40	N	3
89586000	273915100	0.0039	45	N	4
89586000	276301300	0.0031	25	N	3
89586000	289435100	0.0026	35	N	4
89586000	297020500	0.0049	25	N	6
89586000	298853800	0.0029	28	N	4
89586000	311046700	0.0022	71	N	4

Sample Output for M3gl Algorithm					
Query Image	Template Image	Similarity Score	Time Taken	Output	No of Similar Features
89586000	338924500	0.004	28	N	4
89586000	360148100	0.0019	28	N	3
89586000	360536100	0.0027	48	N	3
89586000	378442300	0.0028	44	N	4
89586000	392769200	0.0043	33	N	3
89586000	394660700	0.0038	28	N	3
89586000	402527400	0.0038	54	N	6
89586000	421413300	0.0021	49	N	4
89586000	440687200	0.0032	19	N	3
89586000	465218500	0.0027	49	N	6
89586000	473647800	0.0027	46	N	3
89586000	487974700	0.0036	39	N	5
89586000	493823800	0.0035	75	N	6
89586000	508742400	0.0028	37	N	3
89586000	512680600	0.0023	31	N	3
89586000	525581600	0.0034	22	N	3
89586000	529490700	0.0036	52	N	5
89586000	532264900	0.0031	48	N	5
89586000	550316600	0.0045	48	N	6
89586000	552508800	0.0035	92	N	5
89586000	559172700	0.0036	49	N	7
89586000	575177700	0.0033	21	N	4
89586000	583044400	0.0021	33	N	3
89586000	600485000	0.0039	47	N	3
89586000	616926500	0.002	66	N	3
89586000	622630100	0.0022	35	N	3
89586000	637636000	0.0044	39	N	4
89586000	638402300	0.0016	27	N	3
89586000	660101200	0.0045	27	N	5
89586000	681286000	0.0065	49	N	7
89586000	682285100	0.0031	32	N	4
89586000	682925300	0.0021	127	N	4
89586000	703178900	0.0045	35	N	7
89586000	713053500	0.002	61	N	4
89586000	719445800	0.004	60	N	3
89586000	729960600	0.0062	43	N	9
89586000	733763000	0.0035	59	N	7
89586000	739621800	0.0019	55	N	3
89586000	750049300	0.0026	39	N	4
89586000	755879000	0.0049	35	N	3
89586000	763192800	0.0049	95	N	8
89586000	766985500	0.0034	34	N	5
89586000	774454500	0.0024	24	N	3
89586000	777674900	0.002	49	N	3

Sample Output for M3gl Algorithm					
Query Image	Template Image	Similarity Score	Time Taken	Output	No of Similar Features
89586000	800712400	0.0049	37	N	4
89586000	825689900	0.0029	44	N	6
89586000	852937200	0.002	32	N	3
89586000	869388400	0.005	32	N	5
89586000	871726100	0.0045	32	N	3
89586000	886567100	0.0024	20	N	3
89586000	887682600	0.0025	58	N	5
89586000	890796300	0.0042	33	N	4
89586000	909323300	0.0033	62	N	4
89586000	914735900	0.0062	23	N	3
89586000	935523000	0.0033	37	N	5
89586000	942739800	0.0028	50	N	5
89586000	945029000	0.0042	57	N	8
89586000	977349400	0.0021	25	N	3
89586000	990512300	0.005	25	N	4
89586000	1003423000	0.0029	34	N	3
89586000	1004053500	0.0032	25	N	4
89586000	1035568800	0.0033	21	N	4
89586000	1039875600	0.0023	30	N	3
89586000	1050177000	0.0029	114	N	7
89586000	1062360200	0.003	26	N	3
89586000	1094118000	0.0054	41	N	6
89586000	1118426200	0.0017	239	N	6
89586000	1138980500	0.0047	30	N	4
89586000	1145586200	0.0019	94	N	5
89586000	1153976700	0.0024	35	N	3
89586000	1173929600	0.004	21	N	5
89586000	1183590800	0.0033	44	N	4
89586000	1186403800	0.0032	38	N	3
89586000	1188111000	0.0029	62	N	6
89586000	1206240300	0.0016	52	N	3
89586000	1215261300	0.0021	81	N	4
89586000	1226561800	0.0045	15	N	3
89586000	1226668500	0.0043	41	N	3
111925100	123574800	0.0034	33	N	3
111925100	207140300	0.0036	26	N	4
111925100	222941600	0.0022	36	N	3
111925100	235115100	0.0029	64	N	4
111925100	235318800	0.0016	91	N	3
111925100	248549600	0.0047	28	N	3
111925100	254476300	0.003	44	N	4
111925100	257861600	0.0024	83	N	5
111925100	258269000	0.0018	72	N	3
111925100	258734600	0.0033	31	N	4

Sample Output for M3gl Algorithm					
Query Image	Template Image	Similarity Score	Time Taken	Output	No of Similar Features
111925100	263293600	0.0033	36	N	3

APPENDIX II – Sample Output for MQYW Algorithm

Sample Output for MQYW Algorithm					
Query Image	Template Image	Similarity Score	Time Taken	Output	No of Similar Features
11142100	57479000	0.1329	703	N	3
11142100	89586000	0.1313	710	N	6
11142100	111925100	0.1232	856	N	6
11142100	123574800	0.0975	543	N	1
11142100	207140300	0.1146	481	N	1
11142100	222941600	0.155	704	N	4
11142100	235115100	0.1194	747	N	4
11142100	235318800	0.0933	125	N	4
11142100	248549600	0.2047	511	N	4
11142100	254476300	0.1386	714	N	4
11142100	257861600	0.1249	982	N	8
11142100	258269000	0.0983	45	N	4
11142100	258734600	0.1596	515	N	4
11142100	263293600	0.1065	533	N	3
11142100	267619800	0.1651	850	N	9
11142100	273915100	0.1407	861	N	3
11142100	276301300	0.1965	473	N	3
11142100	289435100	0.1091	558	N	3
11142100	297020500	0.1089	572	N	2
11142100	298853800	0.1897	579	N	3
11142100	311046700	0.1603	56	N	5
11142100	338924500	0.1316	619	N	6
11142100	360148100	0.0958	501	N	2
11142100	360536100	0.1904	590	N	7
11142100	378442300	0.1144	814	N	3
11142100	392769200	0.142	624	N	2
11142100	394660700	0.1173	538	N	1
11142100	402527400	0.0965	717	N	1
11142100	421413300	0.1585	714	N	4
11142100	440687200	0.1592	470	N	3
11142100	465218500	0.1482	619	N	6
11142100	473647800	0.1402	900	N	2
11142100	487974700	0.1134	725	N	3
11142100	493823800	0.0962	840	N	4
11142100	508742400	0.0851	808	N	2
11142100	512680600	0.1223	543	N	3
11142100	525581600	0.1279	480	N	3
11142100	529490700	0.1657	805	N	4
11142100	532264900	0.1459	862	N	9
11142100	550316600	0.1571	793	N	5
11142100	552508800	0.1228	121	N	3

Sample Output for MQYW Algorithm					
Query Image	Template Image	Similarity Score	Time Taken	Output	No of Similar Features
11142100	559172700	0.1368	775	N	6
11142100	575177700	0.1245	456	N	3
11142100	583044400	0.1376	603	N	2
11142100	600485000	0.1367	657	N	3
11142100	616926500	0.1742	841	N	7
11142100	622630100	0.1363	553	N	1
11142100	637636000	0.0919	700	N	2
11142100	638402300	0.1193	575	N	4
11142100	660101200	0.174	389	N	4
11142100	681286000	0.1398	765	N	5
11142100	682285100	0.1223	581	N	3
11142100	682925300	0.0805	549	N	5
11142100	703178900	0.1314	612	N	4
11142100	713053500	0.1405	852	N	5
11142100	719445800	0.0902	39	N	4
11142100	729960600	0.0913	683	N	4
11142100	733763000	0.173	44	N	7
11142100	739621800	0.1402	949	N	3
11142100	750049300	0.0905	512	N	4
11142100	755879000	0.1571	587	N	4
11142100	763192800	0.077	26	N	6
11142100	766985500	0.1885	593	N	3
11142100	774454500	0.1369	477	N	2
11142100	777674900	0.1975	719	N	5
11142100	800712400	0.2011	694	N	3
11142100	825689900	0.1416	698	N	3
11142100	852937200	0.1118	570	N	4
11142100	869388400	0.1116	449	N	4
11142100	871726100	0.1416	474	N	4
11142100	886567100	0.1546	323	N	4
11142100	887682600	0.0748	957	N	3
11142100	890796300	0.1172	513	N	3
11142100	909323300	0.1315	751	N	4
11142100	914735900	0.0962	520	N	2
11142100	935523000	0.104	660	N	1
11142100	942739800	0.1425	666	N	4
11142100	945029000	0.1455	697	N	2
11142100	977349400	0.1534	560	N	5
11142100	990512300	0.1878	471	N	3
11142100	1003423000	0.1519	671	N	2
11142100	1004053500	0.2016	478	N	3
11142100	1035568800	0.1543	458	N	4
11142100	1039875600	0.1715	629	N	3
11142100	1050177000	0.136	85	N	10

Sample Output for MQYW Algorithm					
Query Image	Template Image	Similarity Score	Time Taken	Output	No of Similar Features
11142100	1062360200	0.1486	457	N	1
11142100	1094118000	0.0943	611	N	3
11142100	1118426200	0.065	131	N	4
11142100	1138980500	0.1215	501	N	3
11142100	1145586200	0.1486	270	N	4
11142100	1153976700	0.106	609	N	4
11142100	1173929600	0.1298	481	N	4
11142100	1183590800	0.119	734	N	5
11142100	1186403800	0.1295	691	N	3
11142100	1188111000	0.1356	947	N	4
11142100	1206240300	0.1645	825	N	4
11142100	1215261300	0.0974	95	N	4
11142100	1226561800	0.1034	425	N	1
11142100	1226668500	0.1097	640	N	3
57479000	89586000	0.1661	90	N	10
57479000	111925100	0.1257	3	N	6
57479000	123574800	0.1747	541	N	5
57479000	207140300	0.1563	423	N	7
57479000	222941600	0.2441	634	N	7
57479000	235115100	0.0814	772	N	4
57479000	235318800	0.1638	127	N	10
57479000	248549600	0.1565	515	N	6
57479000	254476300	0.1717	641	N	8
57479000	257861600	0.1167	87	N	6
57479000	258269000	0.1191	960	N	9
57479000	258734600	0.1822	504	N	6
57479000	263293600	0.2024	561	N	11
57479000	267619800	0.0714	704	N	2
57479000	273915100	0.2012	683	N	9
57479000	276301300	0.1718	506	N	8
57479000	289435100	0.1575	551	N	6
57479000	297020500	0.1062	570	N	3
57479000	298853800	0.172	531	N	6
57479000	311046700	0.1665	994	N	5
57479000	338924500	0.1709	538	N	5
57479000	360148100	0.1263	543	N	5
57479000	360536100	0.1645	685	N	7
57479000	378442300	0.1274	684	N	5
57479000	392769200	0.1584	618	N	6
57479000	394660700	0.1488	551	N	3
57479000	402527400	0.1691	745	N	8
57479000	421413300	0.2071	691	N	9
57479000	440687200	0.1033	433	N	4
57479000	465218500	0.1125	639	N	7

Sample Output for MQYW Algorithm					
Query Image	Template Image	Similarity Score	Time Taken	Output	No of Similar Features
57479000	473647800	0.1418	749	N	6
57479000	487974700	0.1699	687	N	10
57479000	493823800	0.1549	789	N	6
57479000	508742400	0.103	730	N	7
57479000	512680600	0.1484	556	N	6
57479000	525581600	0.161	462	N	7
57479000	529490700	0.1441	869	N	9
57479000	532264900	0.172	817	N	12
57479000	550316600	0.1746	929	N	4
57479000	552508800	0.1011	229	N	7
57479000	559172700	0.1658	824	N	11
57479000	575177700	0.1357	537	N	6
57479000	583044400	0.1553	680	N	6
57479000	600485000	0.197	692	N	4
57479000	616926500	0.2021	929	N	10
57479000	622630100	0.1473	602	N	5
57479000	637636000	0.1781	631	N	10
57479000	638402300	0.1962	578	N	5
57479000	660101200	0.14	385	N	4
57479000	681286000	0.194	627	N	7
57479000	682285100	0.1377	561	N	7
57479000	682925300	0.1441	319	N	14
57479000	703178900	0.167	616	N	6
57479000	713053500	0.1165	853	N	10
57479000	719445800	0.129	833	N	9
57479000	729960600	0.1486	629	N	10
57479000	733763000	0.1247	91	N	12
57479000	739621800	0.1923	802	N	10
57479000	750049300	0.1665	568	N	15
57479000	755879000	0.1832	610	N	11
57479000	763192800	0.0759	954	N	9
57479000	766985500	0.183	529	N	6
57479000	774454500	0.1901	532	N	5
57479000	777674900	0.2202	808	N	14
57479000	800712400	0.1611	682	N	3
57479000	825689900	0.1353	642	N	6
57479000	852937200	0.1584	592	N	10
57479000	869388400	0.2114	402	N	16
57479000	871726100	0.1494	459	N	8
57479000	886567100	0.1177	329	N	3
57479000	887682600	0.1504	982	N	5
57479000	890796300	0.1821	523	N	2
57479000	909323300	0.1809	808	N	8
57479000	914735900	0.2069	467	N	10

Sample Output for MQYW Algorithm					
Query Image	Template Image	Similarity Score	Time Taken	Output	No of Similar Features
57479000	935523000	0.1645	661	N	3
57479000	942739800	0.142	701	N	5
57479000	945029000	0.1394	729	N	7
57479000	977349400	0.1712	476	N	4
57479000	990512300	0.1884	445	N	6
57479000	1003423000	0.1614	620	N	5
57479000	1004053500	0.1662	473	N	4
57479000	1035568800	0.1674	486	N	7
57479000	1039875600	0.1384	581	N	5
57479000	1050177000	0.0899	222	N	11
57479000	1062360200	0.1852	495	N	6
57479000	1094118000	0.1355	640	N	7
57479000	1118426200	0.0758	240	N	12
57479000	1138980500	0.1875	491	N	7
57479000	1145586200	0.106	273	N	6
57479000	1153976700	0.2438	601	N	13
57479000	1173929600	0.1989	459	N	5
57479000	1183590800	0.1519	688	N	5
57479000	1186403800	0.1894	637	N	6
57479000	1188111000	0.1594	991	N	9
57479000	1206240300	0.1586	800	N	6
57479000	1215261300	0.1076	135	N	7
57479000	1226561800	0.1617	401	N	4
57479000	1226668500	0.1507	649	N	11
89586000	111925100	0.1282	79	N	12
89586000	123574800	0.1438	676	N	7
89586000	207140300	0.1607	520	N	12
89586000	222941600	0.1617	742	N	9
89586000	235115100	0.1528	879	N	9
89586000	235318800	0.1818	297	N	12
89586000	248549600	0.1435	600	N	10
89586000	254476300	0.1577	777	N	12
89586000	257861600	0.126	410	N	16
89586000	258269000	0.1466	201	N	17
89586000	258734600	0.1371	730	N	13
89586000	263293600	0.1353	725	N	11
89586000	267619800	0.0833	928	N	6
89586000	273915100	0.1823	893	N	11
89586000	276301300	0.1402	594	N	15
89586000	289435100	0.1257	745	N	13
89586000	297020500	0.0975	655	N	10
89586000	298853800	0.1302	649	N	6
89586000	311046700	0.1668	290	N	13
89586000	338924500	0.1151	717	N	12

Sample Output for MQYW Algorithm					
Query Image	Template Image	Similarity Score	Time Taken	Output	No of Similar Features
89586000	360148100	0.1693	657	N	8
89586000	360536100	0.1545	970	N	12
89586000	378442300	0.0889	867	N	6
89586000	392769200	0.163	759	N	11
89586000	394660700	0.1543	643	N	11
89586000	402527400	0.2087	933	N	14
89586000	421413300	0.1751	935	N	20
89586000	440687200	0.1165	515	N	8
89586000	465218500	0.1746	932	N	14
89586000	473647800	0.1963	899	N	10
89586000	487974700	0.1375	871	N	7
89586000	493823800	0.1769	21	N	7
89586000	508742400	0.1055	878	N	9
89586000	512680600	0.1029	697	N	9
89586000	525581600	0.1472	614	N	10
89586000	529490700	0.1535	934	N	13
89586000	532264900	0.1417	931	N	10
89586000	550316600	0.1421	8	N	16
89586000	552508800	0.1261	347	N	18
89586000	559172700	0.177	928	N	12
89586000	575177700	0.125	648	N	7
89586000	583044400	0.1528	793	N	13
89586000	600485000	0.2016	826	N	13
89586000	616926500	0.2003	120	N	16
89586000	622630100	0.157	714	N	10
89586000	637636000	0.1629	779	N	12
89586000	638402300	0.1188	752	N	7
89586000	660101200	0.1276	511	N	9
89586000	681286000	0.1847	804	N	18
89586000	682285100	0.1614	674	N	16
89586000	682925300	0.1762	654	N	18
89586000	703178900	0.1712	750	N	18
89586000	713053500	0.1436	50	N	11
89586000	719445800	0.131	7	N	10
89586000	729960600	0.1197	877	N	9
89586000	733763000	0.1315	226	N	6
89586000	739621800	0.145	29	N	17
89586000	750049300	0.1226	790	N	6
89586000	755879000	0.1642	747	N	9
89586000	763192800	0.1019	154	N	10
89586000	766985500	0.2003	640	N	18
89586000	774454500	0.137	646	N	10
89586000	777674900	0.1475	974	N	10
89586000	800712400	0.1678	790	N	6

Sample Output for MQYW Algorithm					
Query Image	Template Image	Similarity Score	Time Taken	Output	No of Similar Features
89586000	825689900	0.1455	815	N	17

APPENDIX 3 – Sample Codes

a) Function to convert Images from WSQ to BMP

```
private void btnConvert_Click(object sender, EventArgs e)
{
    string caption = "Successful Conversion";
    string Msg;
    int Maxnofile = 0;

    String strfilename;
    String[] wsqFiles;
    MessageBoxButtons buttons = MessageBoxButtons.OKCancel;
    DialogResult result;
    wsqFiles = Directory.GetFiles(txtResources.Text, "*.wsq");
    progressBar1.Value = 0;
    //Path.GetDirectoryName(StrSourcepath);
    Maxnofile = Convert.ToInt32(txtMaxNo.Text);
    Maxnofile = Maxnofile * 4;
    progressBar1.Maximum = Maxnofile;
    for (int i = 0; i < Maxnofile; i++)
    {
        strfilename = wsqFiles[i];
        FileStream fs = File.OpenRead(strfilename);
        byte[] fileData = new byte[fs.Length];
        fs.Read(fileData, 0, fileData.Length);

        WsqDecoder decoder = new WsqDecoder();
        Bitmap bmp = decoder.Decode(fileData);
                    bmp.Save(txtTemplates.Text + "\\" +
      Path.GetFileNameWithoutExtension(wsqFiles[i]) + ".bmp");
        progressBar1.Value = progressBar1.Value + 1;
        //bmp.Save(@"C:\sample_image.bmp");
    }
      Msg = " Convertion Completed Successfully. A total of " + Maxnofile + "
      Files Converted";
    result = MessageBox.Show(Msg, caption, buttons);
}
```

b) Program to perform the matching test

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.IO;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Reflection;
using PatternRecognition.FingerprintRecognition.Core;
using PatternRecognition.FingerprintRecognition.ResourceProviders;

namespace PatternRecognition.FingerprintRecognition.Applications
{
    public partial class frmmatchingtest : Form
    {
        #region private fields

        private MinutiaListProvider mtiaListProvider = new MinutiaListProvider();

        private OrientationImageProvider orImgProvider = new
OrientationImageProvider();

        private SkeletonImageProvider skImgProvider = new SkeletonImageProvider();

        private readonly Dictionary<Type, List<Type>> providersByMatcher = new
Dictionary<Type, List<Type>>();

        private String[] bmpLeftThumbFiles;
        private String[] bmpLeftIndexFiles;
        private String[] bmpRightThumbFiles;
        private String[] bmpRightIndexFiles;

        private int arraysize;

        private double Threshhold;

        private Bitmap qLeftThumbImage;
        private Bitmap qLeftIndexImage;
        private Bitmap qRightThumbImage;
        private Bitmap qRightIndexImage;

        private Bitmap tLeftThumbImage;
        private Bitmap tLeftIndexImage;
        private Bitmap tRightThumbImage;
        private Bitmap tRightIndexImage;

        private Bitmap[] tImagearray;


        private IResourceProvider resourceProvider;

        private ResourceRepository repository;

        private string resourcePath;

        private IMatcher matcher;

        private object qFeatures;

        private object tFeatures;
```

```
        private object[] tLeftThumbFeaturesarray;
        private object[] tLeftIndexFeaturesarray;
        private object[] tRightThumbFeaturesarray;
        private object[] tRightIndexFeaturesarray;

        private object[] qLeftThumbFeaturesarray;
        private object[] qLeftIndexFeaturesarray;
        private object[] qRightThumbFeaturesarray;
        private object[] qRightIndexFeaturesarray;

        Globalvar Globalparams = new Globalvar();

        #endregion

        public frmmatchingtest()
        {
            InitializeComponent();
            var providerByFeatType = new Dictionary<Type, List<Type>>();
            var mtiaListExtractors = new List<Type>();
            var orImgExtractors = new List<Type>();
            var skImgExtractors = new List<Type>();
            var experiments = new List<Type>();
            Assembly thisAss = Assembly.GetExecutingAssembly();
            string dir = Path.GetDirectoryName(thisAss.Location);
            foreach (string fileName in Directory.GetFiles(dir))
                try
                {
                    Assembly currAssembly = Assembly.LoadFile(fileName);
                    foreach (Type type in currAssembly.GetExportedTypes())
                        if (type.IsClass && !type.IsAbstract)
                        {
                            var currInterface =
type.GetInterface("IFeatureExtractor`1");
                            if (currInterface != null)
                            {
                                var featType =
currInterface.GetGenericArguments()[0];
                                if (featType == typeof(List<Minutia>))
                                {
                                    mtiaListExtractors.Add(type);
                                    continue;
                                }

                                if (featType == typeof(OrientationImage))
                                {
                                    orImgExtractors.Add(type);
                                    continue;
                                }

                                if (featType == typeof(SkeletonImage))
                                {
                                    skImgExtractors.Add(type);
                                    continue;
                                }
                            }

                            currInterface =
type.GetInterface("IMatchingExperiment");
                            if (currInterface != null)
                            {
                                experiments.Add(type);
                                continue;
                            }
```

```
                            currInterface =
type.GetInterface("IResourceProvider`1");
                            if (currInterface != null)
                            {
                                var featType =
currInterface.GetGenericArguments()[0];
                                if (!providerByFeatType.ContainsKey(featType))
                                    providerByFeatType.Add(featType, new
List<Type>());
                                providerByFeatType[featType].Add(type);
                                continue;
                            }

                            currInterface = type.GetInterface("IMatcher`1");
                            if (currInterface != null &&
!providersByMatcher.ContainsKey(type))
                                providersByMatcher.Add(type, new List<Type>());
                        }
                    }
                    catch
                    {
                    }
                foreach (var pair in providersByMatcher)
                {
                    var featType =
pair.Key.GetInterface("IMatcher`1").GetGenericArguments()[0];
                    foreach (var provider in providerByFeatType[featType])
                        pair.Value.Add(provider);
                }

                // Populating cbxMinutiaExtractor
                cbxMinutiaExtractor.DataSource = mtiaListExtractors;
                cbxMinutiaExtractor.DisplayMember = "Name";
                cbxMinutiaExtractor.ValueMember = "Name";

                // Populating cbxMinutiaExtractor
                cbxOrientationImageExtractor.DataSource = orImgExtractors;
                cbxOrientationImageExtractor.DisplayMember = "Name";
                cbxOrientationImageExtractor.ValueMember = "Name";

                // Populating cbxMinutiaExtractor
                cbxSkeletonImageExtractor.DataSource = skImgExtractors;
                cbxSkeletonImageExtractor.DisplayMember = "Name";
                cbxSkeletonImageExtractor.ValueMember = "Name";

                // Populating cbxMatcher
                cbxMatcher.DataSource = new List<Type>(providersByMatcher.Keys);
                cbxMatcher.DisplayMember = "Name";
                cbxMatcher.ValueMember = "Name";
            }

            private void cbxMatcher_SelectedIndexChanged(object sender, EventArgs e)
            {

                    switch (matcher.GetType().Name)
                    {
                        case "JY": txtdescription.Text = "Fingerprint Minutiae
Matching Based on the Local and Global Structures";
                            break;
                        case "MJY": txtdescription.Text = "Improved Version of
Fingerprint Minutiae Matching Based on the Local and Global Structures";
                            break;
                        case "QYW": txtdescription.Text = "Fingerprint matching
combining the global orientation field with minutia";
                            break;
```

```
                    case "MQYW": txtdescription.Text = "Improved version of
Fingerprint matching combining the global orientation field with minutia";
                        break;
                    case "PN": txtdescription.Text = "A fingerprint matching using
minutiae triangulation - Using Minutia Triplets";
                        break;
                    case "MPN": txtdescription.Text = "Improved version of
fingerprint matching using minutiae triangulation - Using Minutia Triplets";
                        break;
                    case "MTK": txtdescription.Text = "Improved version of
Fingerprint matching using an orientation-based minutia descriptor";
                        break;
                    case "TK": txtdescription.Text = "Fingerprint matching using
an orientation-based minutia descriptor";
                        break;
                    case "M3gl": txtdescription.Text = "Improved Fingerprint
Verification Using Minutiae Triplets";
                        break;
                }
        }

        private void btnsave_Click(object sender, EventArgs e)
        {
            if (listView1.Items.Count >= 1)
            {
                SaveFileDialog SaveFileDialog1 = new SaveFileDialog();
                string strPath = null;
                SaveFileDialog1.Filter = "CSV Files|*.CSV";
                SaveFileDialog1.FileName = "Data_File";
                SaveFileDialog1.FilterIndex = 1;

                if (SaveFileDialog1.ShowDialog() == DialogResult.OK)
                {
                    strPath = SaveFileDialog1.FileName;
                    ListViewToCSV.SaveListViewToCSV(listView1, strPath, true);
                }
            }
            else MessageBox.Show("No Items to save. The List View is empty");

        }

        private void frmmatchingtest_Load_1(object sender, EventArgs e)
        {
            listView1.View = View.Details;
            listView1.GridLines = true;
            listView1.FullRowSelect = true;
        }


        private void btnRunMatch_Click(object sender, EventArgs e)
        {
            string Templatename = null;
            double SimilarityScore = 0;

            int truepositives = 0;
            int falsepositives = 0;
            int truenegatives = 0;
            int falsenegatives = 0;
            int Totalpositives = 0;
            int Totalnegatives = 0;

            double TPR = 0.0;
            double FPR = 0.0;

            string MatchResult = null;
            string MatchingMinutiae = null;
```

```
            string flt = null;
            int strlength = 0;
            DateTime starttime;
            DateTime endtime;
            Double timetake = new Double();
            double prgval = 0.0;

            string QueryFinger = null;
            string TemplateFinger = null;

            Globalvar GlobalParams = new Globalvar();

            listView1.Items.Clear();
            lblProgressValue.Text = "0";
            progressBar1.Value = 0;
            //set Global parameters
            Globalparams.Resourceprovider = resourceProvider;
            Globalparams.MinutiaProvider = mtiaListProvider;
            Globalparams.orientationProvider = orImgProvider;
            Globalparams.skeletonProvider = skImgProvider;
            Globalparams.Matcher = matcher;
            Globalparams.Repository = repository;

            String path = GlobalParams.Resourcepath;

            if ((path=="") ||(path==null))
            {
                path="C:\\FingerprintImages\\BMPImages";
                Globalparams.Resourcepath = path;
            }

            repository = new ResourceRepository(path);
            bmpLeftThumbFiles = Directory.GetFiles(path, "*" + "_31.bmp");
            bmpLeftIndexFiles = Directory.GetFiles(path, "*" + "_32.bmp");
            bmpRightThumbFiles = Directory.GetFiles(path, "*" + "_36.bmp");
            bmpRightIndexFiles = Directory.GetFiles(path, "*" + "_37.bmp");

            arraysize = bmpLeftThumbFiles.Length;

            //Query  Image array
            qLeftThumbFeaturesarray = new object[arraysize];
            qLeftIndexFeaturesarray = new object[arraysize];
            qRightThumbFeaturesarray = new object[arraysize];
            qRightIndexFeaturesarray = new object[arraysize];

            //Template Image Features
            tLeftThumbFeaturesarray = new object[arraysize];
            tLeftIndexFeaturesarray = new object[arraysize];
            tRightThumbFeaturesarray = new object[arraysize];
            tRightIndexFeaturesarray = new object[arraysize];
            //MessageBox.Show("There are " + arraysize + " Image Files in the
folder " + path);

            // Matching features scores
            List<MinutiaPair> matchingMtiae = null;

            double[] LeftThumbscore = new double[arraysize];
            double[] LeftIndexscore = new double[arraysize];
            double[] RightThumbscore = new double[arraysize];
            double[] RightIndexscore = new double[arraysize];


            //for (double Threshhold = 0; Threshhold < 1; Threshhold += 0.1)
            //{

            Threshhold = 0.5;
```

```
            progressBar1.Maximum = arraysize*2;
            progressBar1.Value = 0;

            IMinutiaMatcher minutiaMatcher = matcher as IMinutiaMatcher;
            //Innitialize Resource Provider

            LoadResources();

            // Commence the experiement
            for (int x = 0; x < arraysize; x++)
            {
                if (matcher != null)
                {
                    string qLeftThumbFileName =
Path.GetFileNameWithoutExtension(bmpLeftThumbFiles[x]);
                    string qLeftIndexFileName =
Path.GetFileNameWithoutExtension(bmpLeftIndexFiles[x]);
                    string qRightThumbFileName =
Path.GetFileNameWithoutExtension(bmpRightThumbFiles[x]);
                    string qRightIndexFileName =
Path.GetFileNameWithoutExtension(bmpRightIndexFiles[x]);
                    QueryFinger = qRightIndexFileName.Substring(0, 10);
                    try
                    {
                        qLeftThumbFeaturesarray[x] =
resourceProvider.GetResource(qLeftThumbFileName, repository);
                        qLeftIndexFeaturesarray[x] =
resourceProvider.GetResource(qLeftIndexFileName, repository);
                        qRightThumbFeaturesarray[x] =
resourceProvider.GetResource(qRightThumbFileName, repository);
                        qRightIndexFeaturesarray[x] =
resourceProvider.GetResource(qRightIndexFileName, repository);
                    }
                    catch (Exception)
                    {
                        MessageBox.Show("Unable to load Query features " +
resourceProvider.GetSignature() + ". Try using different parameters.", "Feature
Loading Error",
                                        MessageBoxButtons.OK,
MessageBoxIcon.Error);
                        return;
                    }
                    for (int i = x+1; i < arraysize; i++)
                    {

                        string LeftThumbFileName =
Path.GetFileNameWithoutExtension(bmpLeftThumbFiles[i]);
                        string LeftIndexFileName =
Path.GetFileNameWithoutExtension(bmpLeftIndexFiles[i]);
                        string RightThumbFileName =
Path.GetFileNameWithoutExtension(bmpRightThumbFiles[i]);
                        string RightIndexFileName =
Path.GetFileNameWithoutExtension(bmpRightIndexFiles[i]);
                        TemplateFinger = RightIndexFileName.Substring(0, 10);
                        try
                        {
                            tLeftThumbFeaturesarray[i] =
resourceProvider.GetResource(LeftThumbFileName, repository);
                            tLeftIndexFeaturesarray[i] =
resourceProvider.GetResource(LeftIndexFileName, repository);
                            tRightThumbFeaturesarray[i] =
resourceProvider.GetResource(RightThumbFileName, repository);
                            tRightIndexFeaturesarray[i] =
resourceProvider.GetResource(RightIndexFileName, repository);
                        }
                        catch (Exception)
```

```
                        {
                            MessageBox.Show("Unable to load Template features " +
resourceProvider.GetSignature() + ". Try using different parameters.", "Feature
Loading Error",
                                            MessageBoxButtons.OK,
MessageBoxIcon.Error);
                            return;
                        }


                        //pbxTemplateImg.Image = tImage;
                        //tImagearray[i] =
ImageLoader.LoadImage(bmpLeftThumbFiles[i]);
                        starttime = DateTime.Now.ToLocalTime();
                        LeftThumbscore[i] =
matcher.Match(qLeftThumbFeaturesarray[x], tLeftThumbFeaturesarray[i]);
                        LeftIndexscore[i] =
matcher.Match(qLeftIndexFeaturesarray[x], tLeftIndexFeaturesarray[i]);
                        RightThumbscore[i] =
matcher.Match(qRightThumbFeaturesarray[x], tRightThumbFeaturesarray[i]);
                        RightIndexscore[i] =
matcher.Match(qRightIndexFeaturesarray[x], tRightIndexFeaturesarray[i]);
                        endtime = DateTime.Now.ToLocalTime();

                        timetake = endtime.Subtract(starttime).Milliseconds;
                        SimilarityScore = (LeftThumbscore[i] + LeftIndexscore[i] +
RightThumbscore[i] + RightIndexscore[i]) / 4;
                        //matchingMtiae = LTmatchingMtiae.Max();// LTmatchingMtiae
+ LImatchingMtiae + RTmatchingMtiae + LImatchingMtiae;
                        //MatchingMinutiae = matchingMtiae.Count.ToString();

                        double Minutiascore =
minutiaMatcher.Match(qRightIndexFeaturesarray[x], tRightIndexFeaturesarray[i], out
matchingMtiae);

                    if (SimilarityScore == 1)
                    {
                        MatchResult = "P";
                    }
                    else
                    {
                        MatchResult = "N";

                    }
                    //Add items in the listview

                    string[] arr = new string[6];
                    ListViewItem itm;

                    arr[0] = QueryFinger;
                    arr[1] = TemplateFinger;
                    arr[2] = SimilarityScore.ToString("0.0000");
                    arr[3] = timetake.ToString();
                    arr[4] = MatchResult;
                    arr[5] = matchingMtiae.Count.ToString();
                    itm = new ListViewItem(arr);
                    listView1.Items.Add(itm);
                    prgval = x + 1;
                    prgval = (prgval / arraysize) * 100;
                    lblProgressValue.Text = prgval.ToString("0");
                    }
                }
                if (Totalpositives != 0) TPR = truepositives / Totalpositives;
                if (Totalnegatives != 0) FPR = falsepositives / Totalnegatives;
            }
```

```
            MessageBox.Show("Experiment Completed", "Experiment",
MessageBoxButtons.OK, MessageBoxIcon.Information,
MessageBoxDefaultButton.Button1);
        }


        private void LoadResources()
        {

            Type resourceType = resourceProvider.GetType();
            foreach (PropertyInfo propertyInfo in resourceType.GetProperties())
            {
                var currInterface =
propertyInfo.PropertyType.GetInterface("IResourceProvider`1");
                if (currInterface != null)
                {
                    var featType = currInterface.GetGenericArguments()[0];
                    if (featType == typeof(OrientationImage))
                    {
                        resourceType.InvokeMember(propertyInfo.Name,
BindingFlags.Public | BindingFlags.Instance | BindingFlags.SetProperty, null,
resourceProvider, new object[] { orImgProvider });
                        continue;
                    }
                    if (featType == typeof(List<Minutia>))
                    {
                        resourceType.InvokeMember(propertyInfo.Name,
BindingFlags.Public | BindingFlags.Instance | BindingFlags.SetProperty, null,
resourceProvider, new object[] { mtiaListProvider });
                        continue;
                    }
                    if (featType == typeof(SkeletonImage))
                    {
                        resourceType.InvokeMember(propertyInfo.Name,
BindingFlags.Public | BindingFlags.Instance | BindingFlags.SetProperty, null,
resourceProvider, new object[] { skImgProvider });
                        continue;
                    }
                }

                if (propertyInfo.CanWrite)
                {
                    currInterface = propertyInfo.PropertyType;
                    if (currInterface.Name == "IFeatureExtractor`1")
                    {
                        var featType = currInterface.GetGenericArguments()[0];
                        if (featType == typeof(OrientationImage))
                        {
                            resourceProvider = orImgProvider;
                            continue;
                        }
                        if (featType == typeof(List<Minutia>))
                        {
                            resourceProvider = mtiaListProvider;
                            continue;
                        }
                        if (featType == typeof(SkeletonImage))
                        {
                            resourceProvider = skImgProvider;
                            continue;
                        }
                    }
                }
            }
```

```
        }
        private void cbxMinutiaExtractor_SelectedValueChanged(object sender,
EventArgs e)
        {
            object selectedValue = ((ComboBox)sender).SelectedItem;
            if (selectedValue != null)
            {
                Type extractorType = (Type)selectedValue;
                mtiaListProvider.MinutiaListExtractor =
Activator.CreateInstance(extractorType) as IFeatureExtractor<List<Minutia>>;
                cbxMinutiaExtractor_Enter(sender, e);
            }
        }

        private void cbxMinutiaExtractor_SelectedIndexChanged(object sender,
EventArgs e)
        {

        }

        private void cbxMatcher_SelectedValueChanged(object sender, EventArgs e)
        {
            object selectedValue = ((ComboBox)sender).SelectedItem;
            if (selectedValue != null)
            {
                Type matcherType = (Type)selectedValue;
                matcher = Activator.CreateInstance(matcherType) as IMatcher;
                Globalparams.Matcher = matcher;

                cbxFeatureProvider.DataSource = providersByMatcher[matcherType];
                cbxFeatureProvider.DisplayMember = "Name";
                cbxFeatureProvider.ValueMember = "Name";

                cbxMatcher_Enter(sender, e);
            }
        }

        private void cbxMatcher_Enter(object sender, EventArgs e)
        {
            propertyGrid1.SelectedObject = matcher;
        }

        private void cbxFeatureProvider_SelectedValueChanged(object sender,
EventArgs e)
        {
            object selectedValue = ((ComboBox)sender).SelectedItem;
            if (selectedValue != null)
            {
                Type providerType = (Type)selectedValue;
                resourceProvider = Activator.CreateInstance(providerType) as
IResourceProvider;
                cbxFeatureProvider_Enter(sender, e);
            }
        }

        private void cbxFeatureProvider_Enter(object sender, EventArgs e)
        {
            propertyGrid1.SelectedObject = resourceProvider;
        }

        private void cbxMinutiaExtractor_Enter(object sender, EventArgs e)
        {
            propertyGrid1.SelectedObject = matcher;
        }
```

```
        private void cbxOrientationImageExtractor_SelectedValueChanged(object
sender, EventArgs e)
        {
            object selectedValue = ((ComboBox)sender).SelectedItem;
            if (selectedValue != null)
            {
                Type extractorType = (Type)selectedValue;
                orImgProvider.OrientationImageExtractor =
Activator.CreateInstance(extractorType) as IFeatureExtractor<OrientationImage>;
                cbxOrientationImageExtractor_Enter(sender, e);
            }
        }

        private void cbxOrientationImageExtractor_Enter(object sender, EventArgs
e)
        {
            propertyGrid1.SelectedObject =
orImgProvider.OrientationImageExtractor;
        }

        private void cbxSkeletonImageExtractor_SelectedValueChanged(object sender,
EventArgs e)
        {
            object selectedValue = ((ComboBox)sender).SelectedItem;
            if (selectedValue != null)
            {
                Type extractorType = (Type)selectedValue;
                skImgProvider.SkeletonImageExtractor =
Activator.CreateInstance(extractorType) as IFeatureExtractor<SkeletonImage>;
                cbxSkeletonImageExtractor_Enter(sender, e);
            }
        }

        private void cbxSkeletonImageExtractor_Enter(object sender, EventArgs e)
        {
            propertyGrid1.SelectedObject = skImgProvider.SkeletonImageExtractor;
        }

        private void cbxMatcher_SelectedIndexChanged_1(object sender, EventArgs e)
        {

        }

        private void cbxFeatureProvider_SelectedValueChanged_1(object sender,
EventArgs e)
        {
            object selectedValue = ((ComboBox)sender).SelectedItem;
            if (selectedValue != null)
            {
                Type providerType = (Type)selectedValue;
                resourceProvider = Activator.CreateInstance(providerType) as
IResourceProvider;
                Globalparams.Resourceprovider = resourceProvider;
                cbxFeatureProvider_Enter(sender, e);
            }
        }

        private void cbxFeatureProvider_Enter_1(object sender, EventArgs e)
        {
            propertyGrid1.SelectedObject = resourceProvider;
        }
    }

}
```